TECHNISCHE UNIVERSITÄT
CHEMNITZ

Fakultät für Informatik

CSR-26-03

Cloud-Based AI for Data Completeness Analysis and Improvement in Predictive Modeling

Sarah Onyinyechi Obasi · Shadi Saleh · Wolfram Hardt

März 2026

Chemnitzer Informatik-Berichte

TECHNISCHE UNIVERSITÄT
CHEMNITZ

Cloud-Based AI for Data Completeness Analysis and Improvement in Predictive Modeling

Master Thesis

Submitted in Fulfillment of the
Requirements for the Academic Degree
M.Sc.

Dept. of Computer Science
Chair of Computer Engineering

Submitted by: Sarah Onyinyechi Obasi
Student ID: 665346
Date: 09.01.2026

Supervising tutor: Prof. Dr. W. Hardt
(further Supervisors) Dr. Shadi Saleh

Abstract

Predictive learning models work best when they are trained on complete and reliable data. In practice, however, many datasets contain missing, inconsistent, or incomplete information. These gaps in data can cause models to make less accurate predictions, introduce bias, and reduce their ability to generalize to new situations.

This thesis introduces a **cloud-based AI framework** that automatically checks data for completeness and improves its quality before it is used in predictive learning. The framework uses the power of cloud computing to handle large and varied datasets. It uses statistical and advanced machine learning techniques to find, measure, and fix missing data. The system also includes features such as real-time data profiling, intelligent methods for filling gaps, and feedback loops that continually improve the data. To build trust, explainable AI tools are used to show how missing data is handled and how this affects prediction results.

Experiments with real-world datasets show that the framework improves both the accuracy and reliability of predictive models while also reducing the time and effort usually needed for data preparation. Because it is cloud-native, the system is scalable, flexible, and easy to integrate into existing data pipelines. Overall, this research shows how cloud-based AI can play a key role in solving the common problem of incomplete data in predictive learning.

Keywords: Data-centric AI, SmartCityCloud, Data Quality Assessment, Data Preprocessing.

Contents

List of Figures

List of Tables

List of Abbreviations

EDA	Exploratory Data Analysis
AI	Artificial Intelligence
VIF	Variance Inflation Factor
RMSE	Root Mean Square Error
AQI	Air Quality Index
IoT	Internet of Things
MLops	Machine Learning Operations
ML	Machine Learning
XGBoost	eXtreme Gradient Boosting
ETL	Extract Transform-Loads
API	Application Programming Interface
GAN	Generative Adversarial Networks
GAIN	Generative Adversarial Imputation Networks
RNN	Recurrent Neural Networks
LSTM	Long Short-Term Memory Networks
GRU	Gated Recurrent Units
KNN	K-Nearest Neighbour
CRISP-DM	Cross-Industry Standard Process for Data Mining
XAI	Explainable AI
DNN	Deep Neural Network
NOAA	National Oceanic Atmospheric Administration
HI	Heat Index
RFR	Random Forest Regressor
DFR	Decision Tree Regressor
MI	Mutual Information
CI	Confidence Interval

1 Introduction

In the current phase of technological transformation across various industries, data has become an important asset on which machine learning relies. With predictive modeling being the backbone of data science, the quality and completeness of datasets play a significant role in obtaining model predictions that are accurate and reliable. However, this is oftentimes not the case with real-world datasets as they contain inconsistent, incomplete and noisy parameters which tamper with the performance of predictive models.

On the other hand, traditional data cleaning and preprocessing methods involve manual checks of missing and incorrect data which does not fit into the complexity of high volume datasets across different sources, as we have today. Thus, the need for a more robust, automated and intelligent framework that can handle the current growing complexity and volume of datasets, cannot be overemphasized.

1.1 Motivation

Artificial Intelligence (AI), over the years, has shown continuous shift towards data-centric methodologies increasing efficiency and data dependency of models. Organizations across industries are harnessing this strategy using data-driven insights to increase efficiency and make better decisions.

In real-world datasets, the most common problem continues to be missing values which can accrue due to human error, integration failures of data, sensor failures, privacy issues, or technical limitations, thereby affecting the reliability and quality of analytical outputs. Predictive models trained on incomplete datasets may exhibit reduced accuracy, biased results, and poor scalability, leading to flawed conclusions and costly decisions. As such, ensuring data completeness is not just a technical requirement but a fundamental prerequisite for trustworthy AI. Cloud computing has introduced new opportunities for processing and managing large amounts of data in real-time, with virtually unlimited storage and compute resources. When AI and cloud infrastructure combine, a strong and flexible platform for building smart systems that can find, evaluate, and improve the completeness of data on their own emerges. A cloud-based AI framework can learn from data patterns, adapt to new data streams, and enhance predictive model performance without extensive manual intervention.

This research explores how cloud-based AI technologies can be harnessed to analyze and improve data completeness in predictive modeling. It presents an integrated framework that automates data completeness assessment, applies intelligent imputation strategies, and evaluates the downstream impact on model accuracy. The study not only contributes to the field of data preprocessing but also offers practical solutions for organizations seeking to utilize AI at scale in cloud environments.

1.2 Problem Statement

Regardless of the continuously increasing importance of data completeness in predictive modeling, there are challenges in the area of establishing a stable, intelligent solution capable of reliably analyzing and improving data completeness and integrating this solution within a cloud-based environment (such as SmartCityCloud). These challenges are caused by the nature and complexity of datasets mostly found within the cloud environment. Existing data preprocessing and imputation methods are typically static, domain-specific, and unable to adapt to the evolving patterns of incomplete data found in large, complex distributed datasets.

This drawback creates a research gap in developing a robust cloud-native AI framework that can not only identify and quantify data incompleteness but also apply intelligent techniques to improve the irregularities in real time while assessing its effect on model performance.

The TUC SmartCityCloud contains a sensor data point, which is made up of a timestamp and a value, stored and managed centrally by the system. In the context of SmartCityCloud, this data forms the basis for creating AI-based data analysis activities. In order to handle, analyze, and improve sensor data for a variety of applications, the current issue is developing a framework that incorporates AI-driven analytics into the system. Workflow design, the use of scalable algorithms, and guaranteeing a smooth interaction with the current infrastructure are the major challenges. Other problems are ensuring that massive amounts of sensor data are processed accurately and effectively, creating AI solutions that are reusable and portable, capable of adjusting to various data sources and finally, delivering lucid outcomes and improved dashboard visualizations for insights that can be put into practice.

To tackle this problem, this research study proposes to design and implement a cloud-based AI framework for data completeness analysis and improvement in predictive modeling, providing automated evaluation, correction, and validation of incomplete data to enhance predictive accuracy of models and AI trustworthiness within the SmartCityCloud.

1.3 What is Data Completeness?

Data completeness simply means having all the required data that should be present in a data set. In other words, a data set contains no missing, empty, or incomplete information for the particular purpose which it is being used[12].
As an illustration, for the Air Quality Index (AQI) data set to be considered complete, all entries of every column and row should be present.

Technically, the degree to which all required data is present in a data set is referred to as data completeness. It asserts that datasets contain all necessary records and attributes needed for accurate analysis, modeling, or decision-making. Incomplete data often stems from missing values, partial records, or loss of information on collection, transfer, or storage of data.

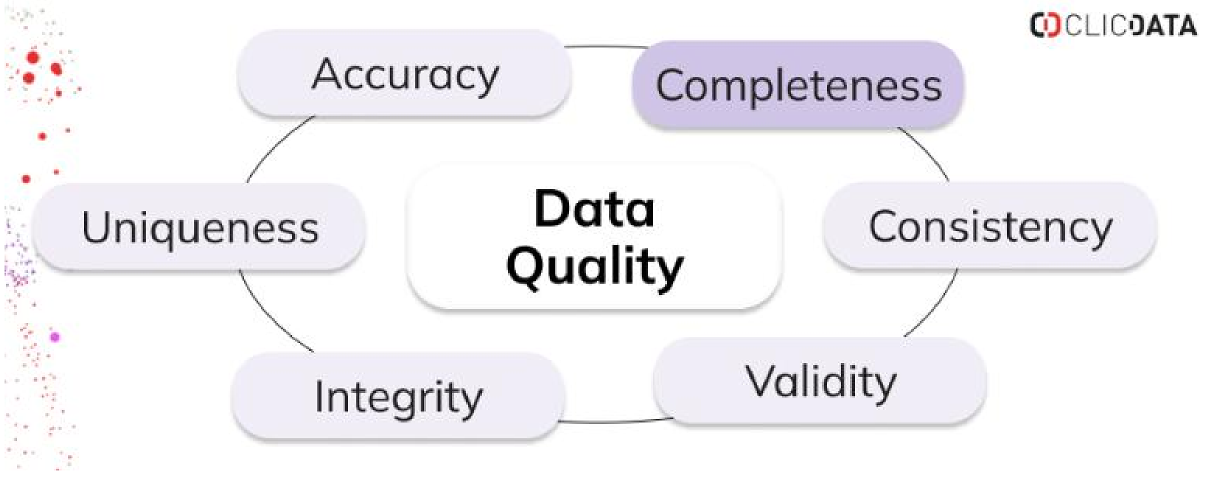


Figure 1.1: Data Completeness, a major factor in Data Quality.[11]

Importance of data completeness

Data completeness is an essential enabler of efficient and reliable data management systems, rather than only a technical need. Its importance transcends several organizational areas, impacting consumer interactions, compliance, operations, and decision-making.

Prediction and Planning: Incomplete data directly impairs the stability of predictive and planning systems by reducing predictive accuracy in model training and validation, introduction of bias and outliers, loss of statistical power as well as poor pattern detection[11].

Operational Efficiency: Utilization of incomplete data can reduce operational effectiveness be it in inventory or supply chain management, leading to delays, errors, lost time and disturbances. Organizations may avoid these errors caused by discrepancies as well as focusing on other higher-value tasks and streamlined processes by giving data completeness top priority.

Reliable Decision-Making: Making well-informed judgement in data-driven fields requires accurate and complete data. Conflicting narratives are produced by erroneous or inconsistent data, which erodes trust in data-driven tactics. For instance, decision-makers may use erroneous tactics that might negatively impact business growth if financial measurements, such as monthly profit, are not consistent with sales and costs[12]. Maintaining consistency reduces mistakes, fosters confidence, and offers a strong foundation for analytical discoveries.

Data Integration and Analysis: For thorough and accurate analysis, organizations are depending more and more on data collected from different sources with different complexity. Nevertheless, irregularities resulting from temporal misalignment or uncontrolled continuous data values may lead to incomplete data causing integration attempts to be more tedious[11]. For instance, AI models trained on incomplete data model identities during data replication lead to null values during analysis, restricting accuracy and useful insights.

1.4 Data Quality in Predictive Modeling

In the era where decision-making is data-driven, the success of predictive learning models whether in business intelligence, healthcare analytic, finance, or engineering largely depends on the quality of the data they are trained on. Predictive learning refers to the process of using statistical and machine learning techniques to model relationships between input variables (features) and an output variable (target) in order to make forecasts or predictions. While advanced algorithms and computational power play crucial roles in determining the accuracy and robustness of these models, better data beats more complex algorithms[13].
Data quality forms the foundation upon which predictive learning systems are built. Poor-quality data can lead to misleading predictions, biased models, and poor generalization performance[14]. Conversely, high-quality data ensures that models

learn meaningful patterns rather than noise. Thus, assessing and improving data quality is a key pre-modeling activity that significantly influences the reliability, interpretability, and ethical soundness of predictive models.
Data quality refers to the degree to which data is accurate, complete, consistent, timely, and relevant for its intended purpose[15]. It is a multidimensional concept that encompasses several measurable attributes:

- **Accuracy:** The closeness of recorded data values to the true values. In predictive learning, inaccurate labels or measurement errors can introduce noise that degrades model performance.
- **Completeness:** The extent to which all required data is present. Missing values can distort feature distributions, reduce statistical power, and introduce bias if the missing values are systematic[14].
- **Consistency:** The absence of contradictions within or across datasets[16]. For instance, a record showing a customer's age as 25 in one database and 52 in another indicates inconsistency.
- **Timeliness:** Data should reflect the current state of the system being modeled. Outdated or stale data may not capture recent trends or behavioral shifts, especially in dynamic environments such as e-commerce or social media analytics.
- **Validity:** Data must conform to defined rules, formats, or domains. Invalid data such as a negative age or out-of-range sensor reading can corrupt feature distributions.
- **Uniqueness:** Duplicate entries can distort sampling distributions and inflate certain observations, especially in imbalanced datasets.

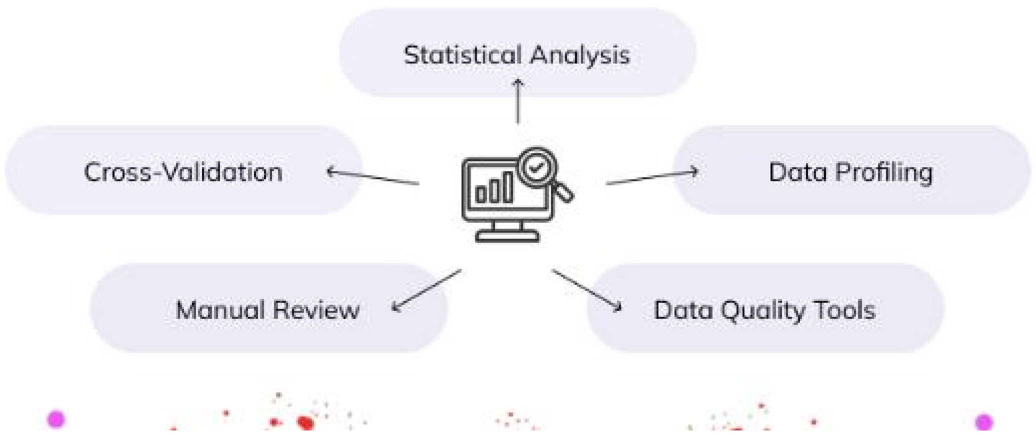


Figure 1.2: Data Validation Methods.[11]

These dimensions determine together how well the data represents the phenomena of interest and thus how effectively a predictive model can learn from it. The relationship between data quality and predictive learning is such that predictive learning algorithms, particularly those based on statistical inference or machine learning, rely on the assumption that the training data accurately represents the real-world system from which it was drawn[14]. When data quality is compromised, the model learns erroneous or biased patterns that may appear statistically valid but fail to generalize.

Common Sources of Data Quality Issues

Data quality problems typically originate from the processes involved in data acquisition, integration, and transformation[3]. Most common sources include:

- **Human Errors:** Manual data entry often leads to typographical mistakes, missing fields, or wrong classifications.
- **Sensor and Instrumentation Errors:** In Internet of Things (IoT) and industrial systems, hardware malfunction or calibration errors can result in unreliable measurements.
- **Integration Issues:** When combining data from multiple sources or databases (as in multi-data-center systems), discrepancies in formats, time stamps, or identifiers can introduce inconsistencies.
- **Data Transmission Failures:** Packet loss, synchronization errors, or latency during data transfer can cause missing or duplicated records.
- **Evolving Data Distributions:** Over time, the statistical properties of data may drift. This is a phenomenon known as concept drift which reduces the relevance of historical data for current predictions.
- **Bias in Data Collection:** Sampling biases occur when certain sub-populations are underrepresented, leading to models that generalize poorly to unseen or minority groups.

Recognizing the source of quality degradation helps design targeted remediation strategies. Data quality influences every stage of the predictive model life cycle from data preparation (determines whether training data adequately captures the target distribution), to deployment and monitoring[2]. Continuous data quality monitoring is essential to detect drift or degradation that may arise after deployment.

In modern MLOps (Machine Learning Operations) frameworks, data quality monitoring is as critical as model monitoring[2]. Automated alerts and retraining triggers can be based on quality metrics such as completeness or drift detection statistics.

1.5 Research Objectives and Significance

This study's main goal is to design, develop, and implement a reliable AI framework for improving data completeness in datasets within cloud-based environments with focus on the TUC SmartCityCloud. In particular, the study seeks to demonstrate that data completeness, as one of the dimensions of data quality, plays a significant role in predictive modeling. By addressing the implications of data completeness, this research aims to achieve the following objectives:

1. **Design an Automated AI-based Solutions for AQI Data Completeness Analysis and Improvement:**
   - The first objective is to develop an efficient AI framework that can assess and improve the completeness of the Air Quality Index (AQI) data set using systematic AI algorithms. This entails finding missing values, outlier detection and removal, duplicate entries, data uniqueness and diversity.

2. **Evaluate these Solutions for Effectiveness, Scalability, and Performance within SmartCityCloud:**
   - In cloud environments like SmartCityCloud, the dispersed nature of data processing and storage presents a significant difficulty. Data completeness is particularly difficult to maintain as data is frequently gathered and processed concurrently across different timestamps. This research focuses on evaluating the suggested frameworks' scalability to make sure they function well in actual cloud environments. The research will evaluate how the frameworks handle enormous amounts of distributed data, guaranteeing their efficacy and scalability for real-time applications. As data quantities and system complexity rise, scalability is crucial to guaranteeing that the frameworks continue to function effectively [3].

3. **Assess the Significance of Improved Data Completeness on AI Model Performance:**
   - To assess the influence on predictive effectiveness of the framework, the model's performance (accuracy, precision, recall, and F1 score) on datasets before and after data completeness enhancements is compared. Comparison of three AI models (Decision Tree, Random Forest and XGBoost) trained on AQI data set to ascertain which model worked best judging by their Root Mean Square Error (RMSE) and R-squared values, will offer factual proof of the suggested strategy's efficacy[10],[2].

One of the most important issues in developing trustworthy AI and predictive analytics systems is ensuring data completeness. When missing values are not random, incomplete data can cause systemic bias in addition to impairing model performance[15]. This research tackles an urgent and frequently disregarded issue in the data science pipeline by putting forth an AI-powered, cloud-native solution. The suggested framework makes it possible to perform automated and scalable data completeness analysis across big datasets and dispersed systems by utilizing cloud infrastructure and artificial intelligence. Improving the completeness of data has a direct impact on improving model training, which raises the predictive models' accuracy, robustness, and diversity[13]. By intelligently automating data preparation and imputation, less manual intervention is required, which lowers operating expenses and deployment time. Although the framework is assessed using particular case studies, its cloud-based design and modular architecture allow it to be used in a variety of industries, including Internet of Things (IoT), e-commerce, healthcare, and finance. An important component of contemporary AI applications is the system's ability to function in dynamic contexts where data changes over time, which is made possible by the integration of continuous learning methods.
In the end, this research advances AI-enabled data quality management and establishes the foundation for more trustworthy and efficient AI systems.

2 Literature Review

The emergence of predictive modeling systems that utilize vast amounts of heterogeneous data is a result of the increasing growth of data in several fields, including enterprise systems, IoT, healthcare, and finance. Cloud computing and AI/ML technologies have also advanced to the point where they allow for flexible, scalable structures for data processing and model deployment. Data completeness, however, remains a major and enduring difficulty, with fragmented stewardship, limited features, inconsistent records, and missing values. Completeness is essential because incomplete or low-quality data weakens predictive model performance, undermines confidence, and can introduce bias[17].

In order to enhance predictive modeling, this literature review looks at how cloud-based AI, that is, AI and ML services and infrastructure hosted and scalable via cloud platforms, can facilitate data completeness analysis. This review explores the importance of data completeness in predictive learning, the role of cloud architectures in AI/ML systems, existing work on completeness analysis, cleaning, and imputation, how cloud services and AI combination is being used in practice and finally, gaps and future directions of data completeness analysis.

2.1 Overview of Completeness as a Data Quality Dimension

Accuracy, consistency, timeliness, validity, uniqueness, and completeness are the main six dimensions or the factors that are oftentimes highlighted in every data quality research[14],[20]. With the growth of data importance lately, data quality dimensions are now extended to conformity, integrity, precision and currency[20]. Specifically, the effects of data quality on Machine Learning performance provides empirical evidence of the relationship between algorithmic frameworks across classification, regression, and clustering tasks and the six quality characteristics or dimensions of data quality, including completeness[17]. These data quality dimensions contend that inaccurate or insufficient training or test data might reduce accuracy, hinder model generalization, and result in untrustworthy judgments.

Challenges in Data Completeness Maintenance: In predictive learning systems, maintaining data completeness is a difficult task impacted by environmental and technical variables especially when dealing with large-scale datasets such as Air Quality Index. Inconsistencies occur when combining data from diverse sources with various formats or sample rates, whereas data gaps are frequently caused by sensor malfunctions, communication breakdowns, or network disruptions. Missing or incomplete records are also a result of operational unpredictability, human mistake, and schema modifications[3],[21]. Additionally, deliberate or inadvertent data omission may result from storage restrictions and privacy laws. These problems may go undiscovered in the absence of strong data governance and monitoring mechanisms, which would eventually lower data quality, model accuracy, and the dependability of predictions from models. Some of the problems include[3][26][28]:

- **Communication Gaps and Sensor Malfunction:** Data loss in IoT-based or sensor-driven systems frequently results from connection problems, power outages, or device faults. For instance, during network outages, air quality sensors may cease sending data, which might lead to incomplete or delayed observations.

- **Diverse Data Sources**: Data is frequently gathered from several platforms, devices, or organizations that employ various formats, standards, and sample rates. Combining such diverse sources results in temporal misalignment, duplication, and inconsistencies that undermine completeness.

- **Time-Series Data Temporal Gaps**: Even brief disruptions in data streams can result in long-term distortions in time-dependent datasets. Trend analysis, seasonality identification, and model training stability may all be impacted by missing values at crucial timestamps.

- **Errors in Human and Manual Data Entry:** Errors, omissions, or delays are frequent when data is partially or completely input by people, as in environmental surveys, labeling, or administrative records. Datasets become systematically incomplete as a result of these human-caused discrepancies.

- **Variability in the Environment and Operations:** The availability and accuracy of readings can be impacted by external factors such as weather, pollution spikes, and sensor calibration drift. Additional gaps may result from instruments automatically discarding data deemed "invalid."

- **Restrictions on Data Storage and Transmission:** Partial uploads or data truncation can occur in cloud-based systems due to bandwidth limitations, network congestion, and storage synchronization problems, particularly during real-time streaming.

- **Security and Privacy Limitations:** Certain data fields may be purposefully left out due to privacy or regulatory restrictions (such as eliminating location or personal identification). Although this procedure is required for compliance, it may lessen the overall completeness of the data.

- **Changing and Adapting Data Schemas**: Datasets may undergo structural changes as systems develop (e.g., new features introduced, old ones discarded). Incomplete feature availability might result from legacy data becoming partially incompatible with new formats.

- **Failures in Data Integration and Pipelines:** Failures or incorrect setups in data ingestion pipelines, ETL (Extract-Transform-Load) procedures, or API interfaces might lead to incomplete data. Such errors frequently go unnoticed until substantial data loss happens in the absence of ongoing monitoring.

- **Limited Governance of Data Quality:** Data completeness is not routinely tracked or maintained in many businesses. Undetected and growing data gaps result from the lack of automated validation criteria, quality dashboards, and accountability frameworks.

Data Completeness Techniques: Ensuring complete and reliable data is essential for accurate predictive learning. To achieve this, organizations combine proactive monitoring with intelligent data recovery and management strategies[24]. Real-time validation and redundant data collection help catch and prevent data loss early, while standardizing data formats and structures reduces inconsistencies across sources. When data gaps do occur, various imputation methods ranging from simple statistical approaches to advanced AI models like deep learning and generative networks, can be used to estimate missing values with higher precision[19]. Data fusion from multiple sensors and feedback-driven refinement further enhance data quality by using contextual information to fill in gaps more effectively[27]. Cloud-based platforms tie these techniques together, offering scalable and automated tools for data cleaning, validation, and integration[25]. Altogether, these methods work to ensure that predictive models are trained on data that is as complete, consistent, and trustworthy as possible.

The Key Role of Cloud-based Frameworks in Data Completeness: Cloud-based frameworks through its elastic computing power, can handle fluctuating data volumes while ensuring that missing, delayed, or corrupted records are automatically detected and managed. One of the main utilities of cloud platforms is centralized data integration since they can allow heterogeneous data sources from IoT sensors, APIs, and databases to be aggregated into unified repositories. Cloud systems also support automated data quality monitoring through AI-based services. Machine learning models deployed in the cloud can detect anomalies, identify missing patterns, and trigger corrective actions in real time[25]. Another major contribution of cloud computing lies in collaborative data management. Through shared, secure access, multiple teams can work on the same datasets while maintaining version control and audit trails. This reduces human-induced incompleteness arising from manual updates or local storage fragmentation [90]. Finally, cloud computing enables context-aware data imputation and enrichment. With the support of distributed AI frameworks, models can be trained across large datasets to predict or reconstruct missing values based on temporal, spatial, and contextual relationships[29].

2.2 Current AI Frameworks for Data Completeness

There are AI frameworks and programming language libraries that can detect missing or incomplete data and handle the inconsistencies as defined by the programmer. These frameworks are a combination of machine learning and deep learning approaches to guarantee reliability of models on complex and time-series datasets [89]. Some of these frameworks include:

- **Time-Series and Sequence Models:** Recurrent neural networks (RNNs), long short-term memory networks (LSTMs), and gated recurrent units (GRUs) are examples of time-series aware frameworks that work well with datasets that have temporal dependencies, like data from air quality or environmental sensors. These mathematical frameworks can learn traits over time and use both past and present observations to guess what values are missing. This makes them good for continuously streaming data[22][23].

- **GAN-Based Frameworks:** Generative adversarial networks (GANs) is one of the deep generative approaches that provides solutions for highly complex datasets with missing or incomplete data. Generative adversarial imputation networks (GAIN) is one of the popular GAN-based frameworks that generates realistic values while maintaining the statistical properties of the original data using imputation[19].

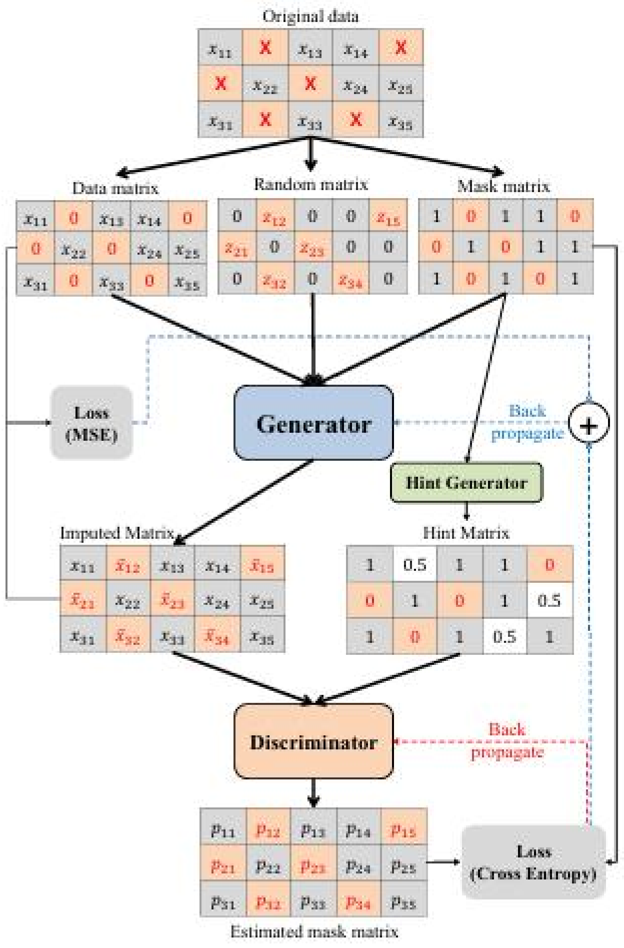


Figure 2.1: Generative Adversarial Imputation Networks (GAIN) Architecture[19].

- **Machine Learning-Based Imputation:** For imputing missing data, conventional machine learning techniques like gradient boosting, random forests, and k-nearest neighbors (KNN) are frequently employed. These methods can handle moderately big datasets with structured features and use correlations between features to approximate missing values. Their performance may deteriorate for extremely complicated or time-dependent datasets, despite their relative simplicity and interpretability[2][24].

- **Cloud-Native AI Platforms:** Cloud computing is used by modern AI frameworks to handle massive amounts of data and carry out real-time completeness analysis. By combining AI-driven imputation, scalable processing, and automated monitoring, these technologies enable businesses to preserve data quality in dynamic settings. Additionally, they improve overall dataset integrity by enabling feedback loops, continuous model updates, and uncertainty quantification[25].

Existing non-AI Frameworks for Data Completeness

Data completeness plays an essential role in predictive models, directly impacting reliability and effectiveness of machine learning models. By controlling how data is gathered, transferred, and processed before it reaches AI models, non-AI frameworks are essential to preserving data completeness. The infrastructure or tools required to effectively analyze and arrange large datasets is provided by programs like AWS Glue, Apache Spark, and Apache Kafka. Large-scale data transformation and cleansing are made possible by Spark[31], dependable real-time data flow is guaranteed by Kafka[30], and cloud-based data integration and cataloging are automated by AWS Glue[32]. Other frameworks that assist in identifying missing or inconsistent data across pipelines include Apache Flink, NiFi, and Great Expectations[33][34][35]. When combined, these technologies provide a solid base that guarantees data is reliable, vetted, and prepared for analysis, freeing up AI models to concentrate on learning rather than correcting defective inputs.

2.3 Current research areas in Data-centric AI

Artificial Intelligence in previous years has focused on building complex models rather than improving the quality of data used to train them. Unlike these conventional model-centric methods, data-centric AI highlights that representative and well selected datasets frequently have a greater impact on model performance than small algorithmic advancements. The goal of current research in this field is to improve the intelligence, cleanliness, and use of data for learning systems across several domains.

Data-centric AI shows better performance with high data quality over conventional model tuning by continuously improving data through feature engineering and error analysis[3], while the goal of model-centric AI is to optimize models that would become the best fit for a given dataset[4] and assumes dataset is complete for training, thereby dataset remains unchanged.

Evaluation and improvement of data quality has remained the major focus of Data-centric AI research. Research focuses on employing automated pipelines that integrate machine learning and statistical techniques to identify, measure, and address problems like missing, inconsistent, incomplete or noisy data [2][38].

Predictive learning applications in fields like healthcare, finance, and environmental monitoring, where data completeness and dependability directly affect model outcomes, stand to gain greatly from such developments.
Efficiency in data labeling and annotation is another area of active research. Reducing human labeling effort while increasing label accuracy and consistency is the goal of strategies like semi-supervised, weakly supervised, and active learning [39]. Large, high-quality training datasets can be created using these techniques even in situations where hand labeling is expensive or scarce.
In order to guarantee transparency, traceability, and compliance in AI pipelines, data governance and provenance have also drawn attention. Data versioning, lineage tracking, and auditing are now supported by cloud-based infrastructures to uphold ethics and legal responsibility across the AI life cycle [40].
Recent research in this field also includes self-healing data pipelines which are self-governing cloud platforms that use reinforcement learning to instantly identify and fix missing data [42], federated and edge completeness analysis that lower latency and privacy risk by verifying completeness at the data source, synthetic data generation producing comprehensive[43], superior synthetic datasets that replicate real-world distributions with the use of generative AI [36], and cloud-native data governance to increase auditability, by incorporating data completeness tracking into lineage graphs and metadata catalogs[41].
Zhong et al. [45] proposed a data-centric strategy to strengthen deep neural network (DNN) models' resistance to harmful perturbations. Their technique enhances model performance against adversarial cases and frequent corruptions by concentrating on dataset augmentation. High scores in a data-centric reliable learning competition showcase the algorithm's performance across different DNN models and show how successful the technique is.
Lastly, to gain a better understanding of how data quality affects model operation and efficiency, researchers are combining explainable AI (XAI) with data-centric techniques to interpret models that provide an explanation for specific corrections or imputations carried out on data[37]. By demonstrating the connection between data integrity and prediction accuracy, this integration boosts confidence in AI results [2].
With all factors considered, contemporary data-centric AI research indicates a larger shift in AI development, shifting from enhancing algorithms to enhancing the data foundation itself. This change promotes AI systems that are more transparent, dependable, and scalable, especially in cloud-based settings where predictive learning heavily relies on entire data[1][4].

2.4 Gaps in Ongoing Research Areas

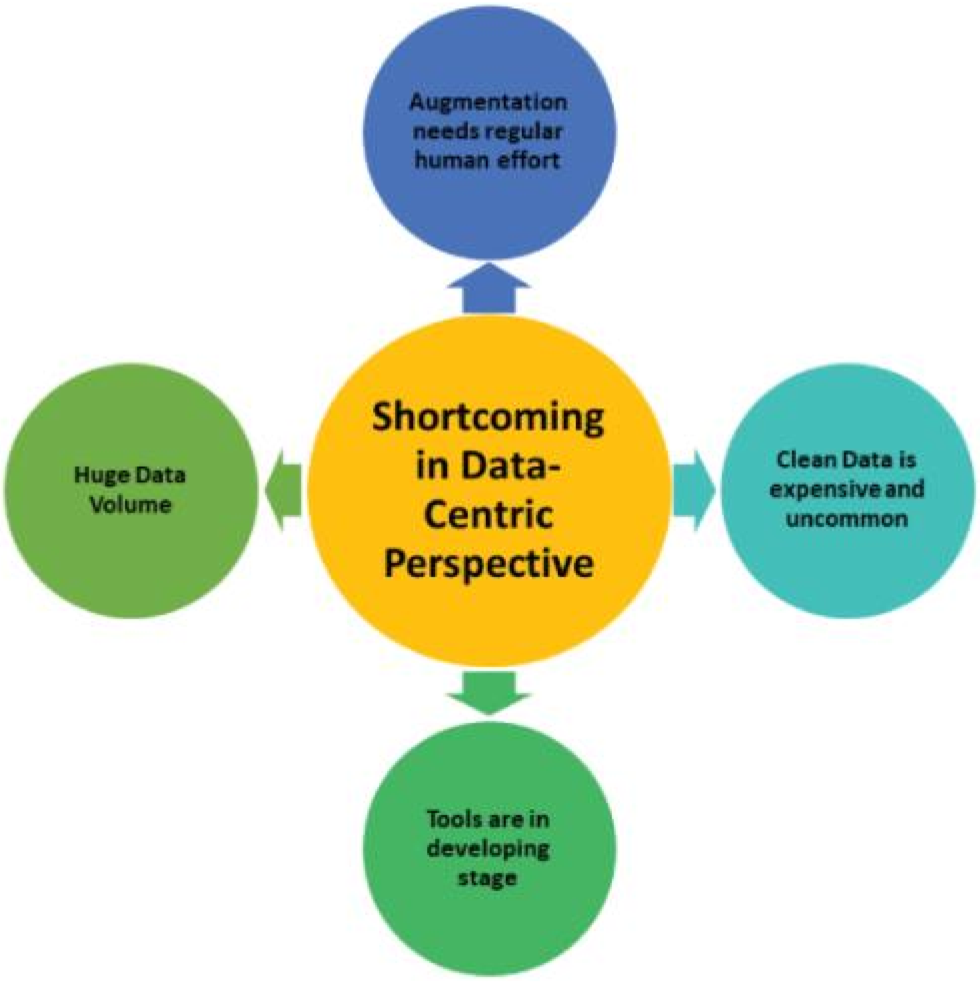

Augmentation needs regular human effort
Huge Data Volume
Shortcoming in Data-Centric Perspective
Clean Data is expensive and uncommon
Tools are in developing stage

Figure 2.2: Challenges of Data-Centric AI Systems[44].

Lack of Robust Frameworks that combines both Data-centric AI and Model-centric AI

Current systems lack systematic fusion of both AI approaches for the improvement of both model and data quality. Using both approaches will enhance models using traditional optimization techniques while employing data-centric principles for robust data engineering. This considers data and model as interwoven gears that provide remarkable outcomes, as neatly suggested in [4]. By appreciating the achievements of earlier work, this embracing of synergy respects ethical research while acknowledging the current state of affairs. Consistency, sufficient volume retention following cleaning, maintenance quality, the need for an appropriate data versioning system, etc. are some of the obstacles that data-centric manufacturing must overcome. The loss of volume related to data cleansing and validation is one of the most important of the many problems. Only when low-quality datasets are eliminated can an AI model be fed a vast amount of high-quality data[44].

Real-Time Processing and Monitoring Issues

Many data-centric AI applications depend on real-time data streams, including autonomous driving, financial trading, sensor monitoring, and social media analytics. These scenarios demand rapid data processing and near-instant model predictions, yet maintaining accuracy under high-velocity conditions remains difficult. Existing systems often lack the computational efficiency and robustness required for continuous, time-sensitive inference. Therefore, more advanced and scalable algorithms for real-time data processing are needed to support the demands of modern data-centric AI systems [46].

Under-explored Human-AI Collaboration for Data Completeness

Increasingly efficient ways for people and AI systems to work together are required as AI systems become increasingly integrated into different fields. Human-AI connection is fraught with difficulties, including simplicity, accessibility, and trustworthiness. This entails figuring out how to make AI systems more understandable and visible to people as well as creating means for people to provide AI systems with input and direction[47].

Limited Focus on Data Completeness-Specific Frameworks

Making sure the data needed for both training and testing AI models is complete and of high quality is one of the primary problems in data-centric AI. Incomplete, or biased data may cause AI models to perform poorly or make incorrect predictions. Although it can be costly and time-consuming, thorough data preparation is an essential stage in the AI pipeline which often is isolated as a variable affecting predictive learning. This covers correcting missing or faulty data as well as cleaning, normalizing, and converting data[44].

Limited Explicit Studies on Data Completeness in Cloud AI Pipelines

Although data quality and big data analytics literature highlight incompleteness, there are few empirical studies specifically focused on “cloud‑based AI systems for completeness analysis” and measuring how completeness improvement affects predictive outcomes in cloud settings[48].

2.5 Overview of SmartCityCloud Architecture

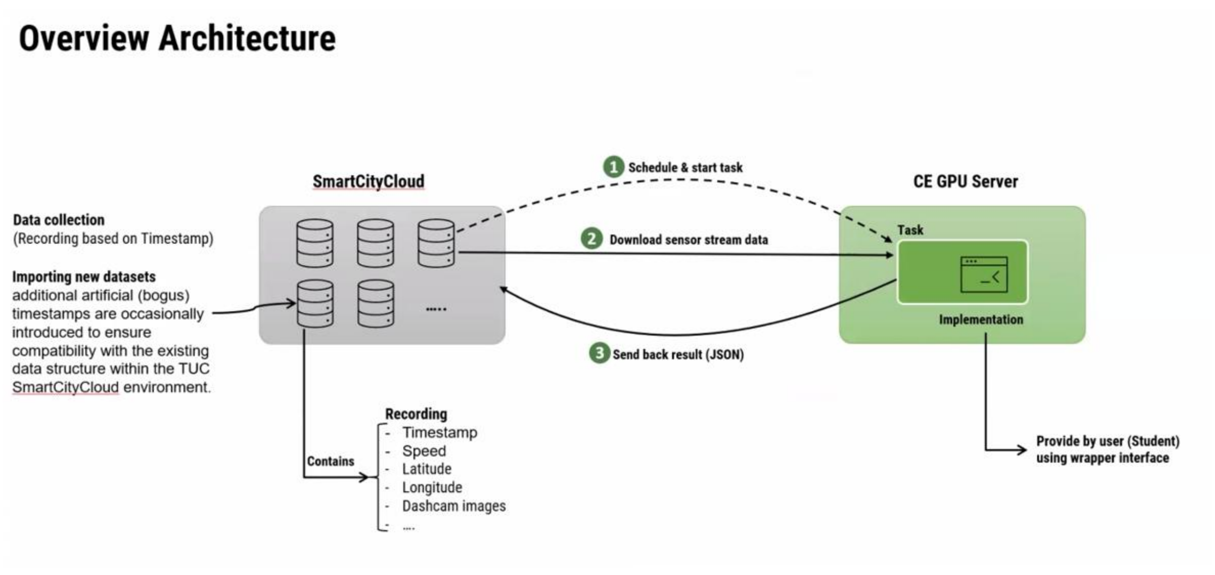

Overview Architecture
SmartCityCloud
CE GPU Server
1 Schedule & start task
2 Download sensor stream data
3 Send back result (JSON)
Task
Implementation
Data collection
(Recording based on Timestamp)
Importing new datasets
additional artificial (bogus) timestamps are occasionally introduced to ensure compatibility with the existing data structure within the TUC SmartCityCloud environment.
Contains
Recording
- Timestamp
- Speed
- Latitude
- Longitude
- Dashcam images
- ....
Provide by user (Student) using wrapper interface

Figure 2.3: Overview of SmartCityCloud System Architecture.

A date and a value make up each data point in the TUC SmartCityCloud system, which is a centralized platform for managing and storing sensor data records. The SmartCityCloud environment uses this data as the basis for creating AI-based data analysis jobs. Developing a framework that incorporates AI-driven analytics into the system to process, evaluate, and improve sensor data for diverse applications is the current problem. Workflow design, scalable algorithm implementation, and smooth connection with the current SmartCityCloud infrastructure are among the tasks of this research. The architecture of the TUC SmartCityCloud system is divided into 3 primary components:

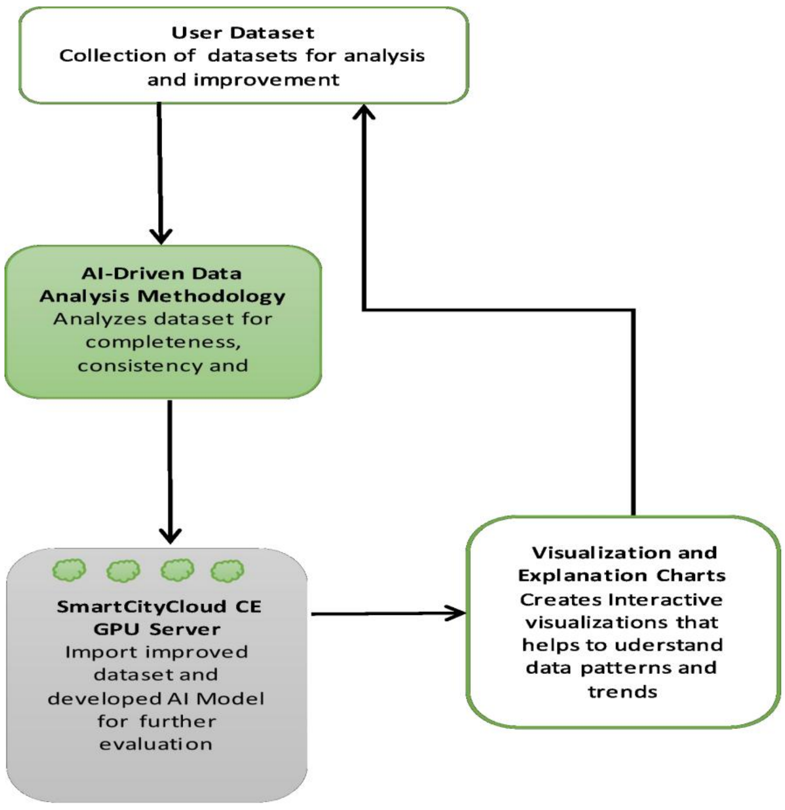
SmartCityCloud Server: This component serves as the central repository for sensor data collection. Each dataset in the system is represented as a series of tuples, where each tuple contains timestamps and associated sensor values. To facilitate advanced data analytics, artificial intelligence (AI) methods are applied to these datasets. However, when importing new datasets, additional artificial timestamps are occasionally introduced to ensure compatibility with the existing data structure within the TUC SmartCityCloud environment.

CE GPU Server: This server acts as the primary computational unit for executing AI-driven methods. It is designed to handle tasks requiring significant processing power, such as advanced analytics or machine learning model execution.

Web Interface: The web interface of the SmartCityCloud which is separate from the server serves as a friendly user interface for visual interactions starting from the importation of the dataset to the visualization of results through the dashboard.

SmartCityCloud Task Workflow

The workflow for computing tasks on the TUC SmartCityCloud simply consists of dataset selection and upload, task initialization and selection, scheduling of tasks and transmission of results on task completion.



User Dataset
Collection of datasets for analysis and improvement
AI-Driven Data Analysis Methodology
Analyzes dataset for completeness, consistency and
SmartCityCloud CE GPU Server
Import improved dataset and developed AI Model for further evaluation
Visualization and Explanation Charts
Creates Interactive visualizations that helps to uderstand data patterns and trends

Figure 2.4: SmartCityCloud Task Workflow.

3 Methodology

The main goal of this thesis is to increase data quality by improving data completeness so that machine learning models can perform better in terms of prediction. The dependability of model predictions can be compromised by poorly organized, inaccurate, or incomplete data, which is a serious problem in data-driven applications where precise results are necessary for making decisions. This study presents a structured process that methodically cleans, verifies, and enhances the dataset before model construction in order to overcome this problem. For analytical and predictive applications, the goal is to guarantee that the processed data is reliable and correct.

3.1 Research Design and Approach

The overall research approach aligns with the Cross-Industry Standard Process for Data Mining (CRISP-DM), as it follows structured stages of problem definition, data assessment, completeness improvement, model development, evaluation, and deployment planning [49]. This work employs a quantitative, observational research approach focused on the development and comparison of supervised learning models for prediction, utilizing data obtained from routine meteorological sensor observations. The overarching purpose is to construct a robust regression modelling framework that remains stable even under realistic challenges such as missingness, outlier presence, correlated predictors, and sensor inconsistencies. The Cross-Industry Standard Process for Data Mining (CRISP-DM), a popular paradigm that describes an organized lifecycle for analytics projects, serves as the foundation for this method. Business understanding, data understanding, data preparation, modeling, assessment, and deployment are among the interrelated, iterative steps that CRISP-DM breaks down the workflow into[49][50].

- **Business Understanding:** This phase focuses on the comprehension of the issue that has to be resolved from a business angle and translating it to a data mining problem definition and resolving plan.
- **Data Understanding:** The data is the accessible raw material that will be used to build the solution. Understanding the data quality structure happens in this phase.
- **Data Preparation:** This involves manipulating and transforming the data into formats that produce better outcomes.
- **Modeling:** This focuses on building a model or pattern that captures irregular patterns in the data.

- **Assessment**: Before deploying the data mining findings, they must be thoroughly evaluated to ensure their validity and dependability.
- **Deployment:** This is simply the use of an evaluated predictive model in an information system or business process and is usually the outcome of data mining.

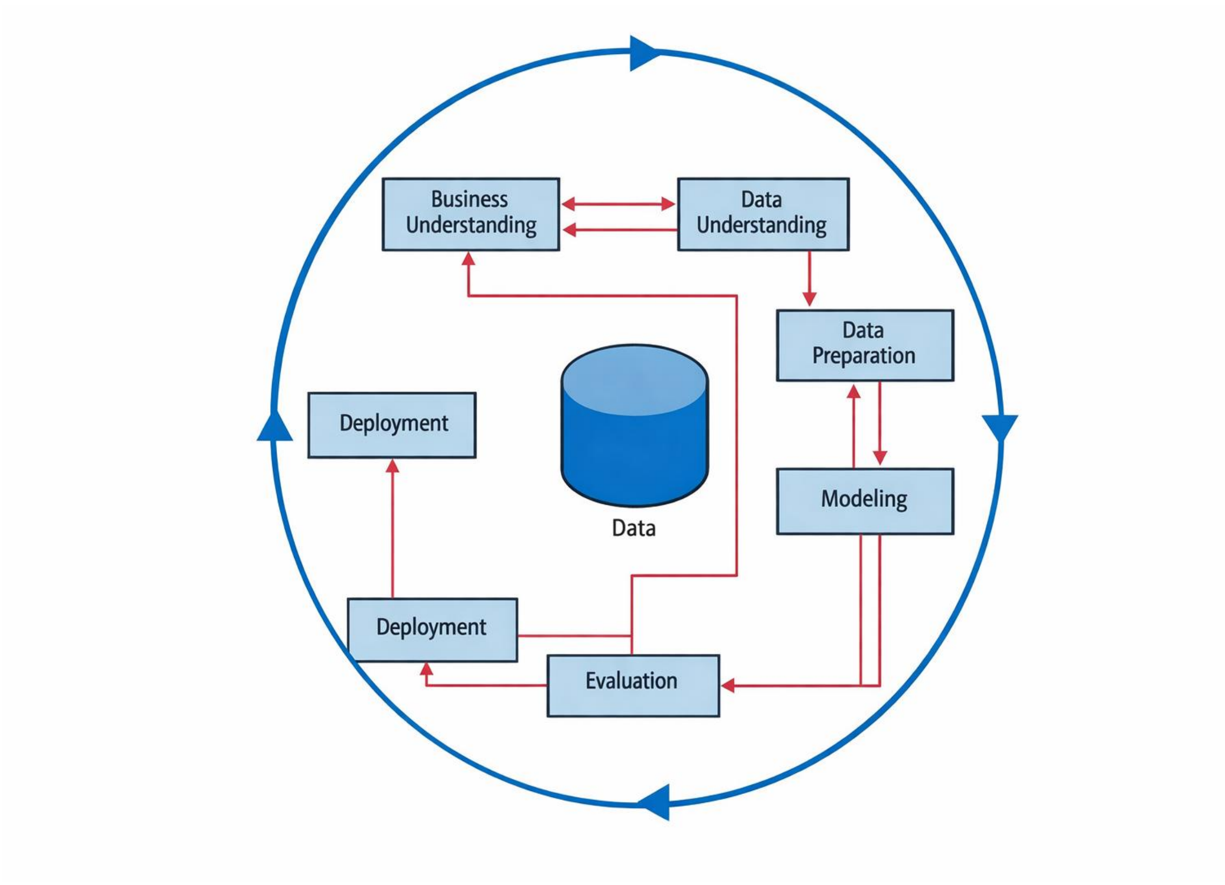


Figure 3.1: Cross Industry Standard Process for Data Mining[49].

The workflow implemented in this study begins with preliminary preprocessing, during which the dataset is loaded, standardized, and subjected to initial cleaning to establish a complete baseline. Once the dataset passes these completeness validation checks, Exploratory Data Analysis (EDA) is performed to examine attribute distributions, detect outliers, and identify underlying relationships. EDA checks revealing patterns and irregularities that may not be visible through rule-based validation alone. To evaluate the impact of data completeness on model performance, the cleaned dataset produced through this workflow is used to train three machine learning models for performance comparison using the Random Forest algorithm, Decision Tree algorithm and XGBoost, selected for their robustness, suitability for tabular data, and strong predictive capabilities. Model performance is evaluated using standard regression metrics, including R-squared values and Root Mean Squared Error (RMSE).

The contribution of this research lies in its systematic approach to data quality management and its demonstration of the measurable benefits of completeness-focused preprocessing. By improving data completeness prior to model training, it is anticipated that the Cleaned Dataset will yield superior predictive performance compared to the Raw Dataset. The findings of this study are expected to reinforce the critical role of data quality in machine learning and to provide a reproducible framework that can be applied across diverse datasets and domains where reliable predictive modeling is required. The result is a methodological approach that is rigorous, reproducible, and well-aligned with contemporary standards in data-driven meteorological modelling.

3.2 Data Cleaning and Preprocessing

Data cleaning and preprocessing form a critical foundation in this study because the accuracy, reliability, and stability of any predictive model are directly determined by the quality and completeness of the input data. Environmental and meteorological sensors are inherently complex due to varying instrument characteristics, constantly changing weather conditions, and external physical disturbances. As highlighted by Slater in [51], raw meteorological readings often contain irregular spikes, sudden jumps, and abnormal fluctuations at the sensor level. These distortions must be identified and corrected before applying statistical or machine learning techniques. Therefore, this section outlines the comprehensive preprocessing and data refinement procedures applied to the Air Quality Index (AQI) datasets, ensuring they are scientifically valid, consistent, and analytically suitable for cloud-based predictive learning.

3.2.1 Data Source and Unit of Analysis

The dataset used in this study was collected from SmartCityCloud, a platform that stores continuous records of atmospheric conditions at fixed time intervals from monitoring stations as well as other meteorological data as discussed in Chapter 2. The station operates autonomously using embedded sensing equipment designed to measure key environmental variables such as air temperature, humidity, wind speed, wind direction, solar radiation, and rainfall.

This raw dataset contains more than 28,000 individual observations and includes 12 variables, each associated with a meteorological or sensor-driven phenomenon. Core attributes include: AirTemperature, Humidity, SRAD, SRADCumulative, WindSpeed, WindDirection, Rain, BarometricPressure, as well as time- and location-specific metadata such as Latitude, Longitude, CollectedDateAt. Several engineered features are also produced during preprocessing, while additional technical indicators, such as GPSFix, verify whether the station was functioning correctly at the time of each measurement.

Each record corresponds to a time-stamped sensor reading, representing a discrete snapshot of environmental conditions. Treating each time-stamped measurement as the unit of analysis is particularly suitable for predictive modelling, as it allows the model to learn both immediate and evolving patterns within atmospheric dynamics. Because the dataset spans an extended time period, it exhibits characteristics of a cross-sectional time series, enabling the detection of temporal trends, diurnal cycles, weekly patterns, lag relationships, and seasonal effects. This temporal continuity enhances model effectiveness but also introduces practical challenges such as irregular sampling intervals, sensor dropouts, missing readings, and data clustering—all of which must be addressed through data cleaning and completeness evaluation.

From an ethical standpoint, the dataset presents minimal risk, as it contains no user-generated or personally identifiable information. All data reflect ambient environmental conditions recorded automatically for monitoring purposes. The processing workflow used in this research is fully reproducible, and the original dataset remains unaltered.

Overall, the dataset's size, temporal resolution, diversity of environmental variables, and sensor precision make it a strong candidate for developing machine learning models aimed at predicting barometric pressure with scientific reliability and analytical robustness.

3.2.2 Dataset Analysis and Cleaning Steps

Before applying any cleaning procedures, it is essential to examine the dataset in detail to understand its measurement characteristics, and potential irregularities. Therefore, the initial stage involves inspecting variable types, ranges, units, formats, and statistical distributions. Profiling of the dataset are performed using common Python functions such as info(), describe(), and value_counts() to assess completeness and detect early anomalies. First, the timestamp column (CollectedDateAt), are standardized to ensure a consistent time representation, preventing discrepancies caused by differing formats or implicit time zones. Initial distribution checks revealed long-tailed behaviours and sporadic spikes across several variables—patterns commonly associated with atmospheric observations due to environmental noise or sensor instability. Missing values are quantified using isnull().sum(), and further explored with missingness heatmaps. Overall, most variables showed either no missing records or minimal missingness.

Outlier detection with summary statistics to locate extreme deviations are followed by visual inspection using boxplots, time-series plots, histograms, and scatterplots. In order to mitigate distributional distortions, the extreme bottom and top 1% of BarometricPressure variable values are winsorized. Winsorization is a technique designed to suppress the influence of extreme outlier values while allowing the data to still be used. A comparison of the pre and post winsorization histograms shows that the distribution tail effects have been reduced and the distribution symmetry has improved, confirming a better performance of the model which is not influenced by rare extreme values located on the periphery of the distribution. Winsorization is selected to improve robustness while maintaining scientific validity by capping values above the 99th percentile at that percentile, and values below the 1st percentile were capped likewise[52]. This approach reduced the influence of sensor artefact while still preserving natural atmospheric variability.
Wind direction and speed are converted into orthogonal components (Wind_u and Wind_v) to resolve issues of circularity—for example, 359° and 0°, which appear numerically distant but are meteorologically adjacent. Converting wind data to vector components, as standard in atmospheric modelling, allows predictive models to better capture pressure responses to wind behaviour[53]. Normality checks processes skewness and kurtosis analysis to confirm long-tailed distributions which are typical of real environmental datasets influenced by climatic extremes.

3.3 Data Completeness Evaluation Criteria

Defining clear data completeness evaluation standard is essential for ensuring that datasets used in artificial intelligence (AI) applications are reliable and suitable for analysis. Completeness criteria provide the basis for defining what constitutes clean, complete and usable data within a given context, and they help regulate the structure, format, and semantic meaning of the information being processed. This is particularly important in cloud-based platforms such as SmartCityCloud, where heterogeneous datasets must remain uniform in order for AI models to produce accurate and reliable outputs. Within this framework, the importance of data completeness serves as a foundational requirement addressed when preparing environmental data for predictive modelling, especially in automated sensing systems where instruments operate continuously under varying conditions. These systems are inherently prone to communication interruptions, mechanical failures, and measurement noise originating from meteorological variability. Environmental datasets are time-dependent and influenced by multiple interacting physical phenomena, making them inherently complex. Therefore, datasets must not only contain adequate records but must also be structurally coherent and contextually informative before machine learning methods can be applied. Hence, the definition of evaluation criteria of the dataset according to several completeness dimensions: attribute completeness, value completeness, schema completeness, time completeness, and category completeness. These dimensions collectively align with data quality expectations in modern data mining workflows such as the CRISP-DM process [49][50]. In this context, completeness does not simply imply the absence of missing values rather a dataset is considered complete only when it provides sufficient information. The evaluation conducted here ensures that the predictive learning tasks in Model training are built upon data that is representative and scientifically meaningful.

3.3.1 Attribute Completeness

Attribute completeness refers to the extent to which all required attributes (or data fields) in a dataset are present and adequately populated. In the context of AQI dataset, attributes correspond to specific variables such as temperature, humidity, wind speed, barometric pressure, and rain. A dataset is considered to have high attribute completeness when all expected attributes necessary to describe the physical process under study are included and recorded across the dataset.

Unlike simple checks for missing values, attribute completeness evaluates whether the dataset contains all the essential variables needed to characterise the meteorological phenomena being modelled.

Key Aspects of Attribute Completeness include[54]:

- **Presence of Values:** Values should exist without gaps or missing values in each attribute. This is to ensure that model's understanding of the environment or hidden relationships remains undistorted.
- **Degree of Missingness:** This quantifies how much information is missing (e.g., 4% vs 55%) and guides decision making. For example, imputation can be used in filling small gaps while large gaps can be dropped or the attribute totally replaced.
- **Pattern of Missingness:** This Investigates whether missing data is random or has a pattern (e.g., rain sensor fails only during storms). These Patterns reveal sensor faults or biased measurements, which affect predictions.
- **Consistency Across Observations:** Ensures the attribute uses uniform units, formats, and ranges over time because inconsistent units (e.g., Celsius vs. Kelvin) can destroy model accuracy.
- **Representativeness:** This determines whether the available values reflect the true physical phenomenon being measured. For example, even if values

exist, they can be incomplete scientifically (e.g., no night-time solar radiation readings).

Importance of Attribute Completeness:

The accuracy, integrity, and utility of AI predictions are directly impacted by attribute completeness. The AI model may discover false relationships, make erratic or skewed forecasts, or misunderstand the behavior of the environment (e.g., forecasting rainfall using inaccurate humidity values), if the attributes (columns) are lacking values or don't fairly depict the process under observation. In essence, attribute completeness answers the question: “Are all relevant attributes present to describe the environmental process?” Ensuring attribute completeness is a prerequisite for subsequent data quality checks, including value accuracy, schema consistency, temporal continuity, and category validity. As part of the CRISP-DM framework, this check ensures that the data supports the analytical objectives defined during the business understanding and data understanding phases [50].

Example: An AI model for predicting rainfall intensity in SmartCityCloud may not be effective if attributes related to humidity or air pressure are absent in the dataset, even if the existing records have no missing values. Therefore, attribute completeness determines whether the dataset provides a sufficiently comprehensive basis for analytical modelling.

3.3.2 Value Completeness

Value completeness which can also be called validity completeness ensures that the data within a dataset is correct, logically coherent, and falls within expected natural limits or set standards. Machine learning models may become less predictive as a result of noise or bias introduced by incomplete values, such as outliers, duplicates, or incorrect entries [55]. For instance, an obvious value mismatch occurs when a temperature sensor registers an implausible measurement like -5000°C for AirTemperature attribute in a dataset. In a similar vein, missing values or duplicate entries might skew model training and jeopardize the accuracy of results.

Key Aspects of Value Completeness:

- **Value Ranges Check:** Each attribute in a dataset should have a defined range of values in which the values of the attribute should fall in, and any value outside this range should be flagged as incompleteness. This is to eliminate anomalies that can introduce inaccuracies into AI models and decrease the reliability of model decisions [18].

- **Values Incontradiction:** To ensure logical coherence within the dataset, values across linked data fields should not conflict with one another. AI algorithms may be misled by conflicting entries, resulting in incorrect forecasts or judgments.
- **Value Accuracy and Validity:** In accordance with the intent and context of the dataset, the values should be legitimate and appropriately represent the phenomenon being measured. Unreliable model outputs might result from training data corruption caused by invalid or incorrect entries [55]. For instance: for a rainfall attribute, only numerical values should be expected as a valid complete value.

Importance of Value Completeness:

Maintaining value completeness is critical for ensuring that data is reliable and interpretable. Incomplete or out-of-range values can cause machine learning models to misinterpret information, thereby making false or inaccurate predictions. For example, value incompleteness can occur when attributes go beyond the limits of their natural occurrences(Humidity: -0.0087%)
Such inconsistencies disrupt model training and reduce predictive accuracy, highlighting the necessity of enforcing value completeness across datasets.

3.3.3 Time Completeness

Time completeness refers to the extent to which a time-series dataset contains all expected observations within a defined temporal interval[84]. For weather, climate, and other automated sensing systems, measurements are typically recorded at fixed intervals (e.g., every 1, 5, or 10 minutes). Time completeness ensures that the dataset has no missing timestamps, irregular gaps, or inconsistent recording frequencies. In simple terms, a time-series dataset is time complete if it fully represents the intended measurement period without any missing or duplicated time entries.

Key Aspects of Time Completeness:

- **Presence of Timestamps:** All expected time values must exist and dataset timestamps are included with no missing data for any day, week or month because missing timestamps disrupt continuity and prevent correct trend modeling.
- **Duplicate Timestamps:** Duplicate timestamps introduces redundancy in model training, causing double counting and distortions in statistics. Every

field should be checked for timestamp duplication. For example, any timestamp recorded more than once.

- **Identification of Time Gaps:** For instance, detection of missing blocks or long periods of intervals without data records across fields.

Importance of Time Completeness:

There are complex time-dependent datasets across domains today. Incomplete time logs often leads to misleading trend analysis (e.g rainfall change detection), errors in model training, incorrect event detection or even incorrect averages or statistical summaries. Ensuring that datasets are time complete is critical before applying forecasting or predictive models, data fusion and anomaly detection[84].

3.3.4 Schema Completeness

The goal of schema completeness is to preserve consistency in the dataset's structure or design. It guarantees that every item in a dataset adheres to a predetermined schema, which includes data types, field order and uniform metadata units. Dataset simply maintains thesame uniform data structure in all fields. The creation of well-structured datasets, that are easier for machine learning models training and other data processing tools to handle, depends greatly on schema standardization.

Key Aspects of Schema Completeness:

- **Data Structure Uniformity:** Every field in the dataset should follow a certain data format. For instance, all records must store this data as strings if the Location column is designated as a string. contradictions would be introduced if information were stored as integer or decimals, which might result in inaccuracy during data analysis [83].
- **Field Standardization:** Each entry in the dataset must have the same fields or characteristics. For example, every entry in a the AQI dataset should completely contain variables like BarometricPressure, AirTemperature, Rain and CollectedDate. This uniformity guarantees that no crucial information is absent from any record and removes uncertainty [81].
- **Field Structure and Labeling:** All fields in the dataset must have the same order and name. This reduces the possibility of errors during integration or analysis and guarantees predictability. For example, since these variations might interfere with automated procedures, a field should remain exactly as named across other records [80].

- **Declaration of Compulsory Fields:** A number of the dataset's fields are required and cannot be left out of any record. For instance: in a weather dataset like AQI, attributes like BarometricPressure and AirTemperature are of key importance. The data would be greatly incomplete and inappropriate for effective analysis if these fields were to be missing [82].

Importance of Schema Completeness:

In order to guarantee the authenticity and usefulness of datasets in AI systems, schema completeness is essential. During data preparation and development of models, nonuniform schemas such as missing fields, wrong data types, or misaligned field orders can result in catastrophic issues. Schema completeness can lead to training errors and incorrect predictions since machine learning models depend on structured and predictable inputs.

3.3.5 Category Completeness

Category completeness evaluates whether all expected classes or categories are present, valid, and correctly recorded in a dataset[84]. In IoT and meteorological systems, this applies to sensor status flags (e.g., GPSFix, operational/error codes), wind direction bins (N, NE, S, etc.), rain/no-rain event classes and not limited to quality control labels. It ensures that all valid classes exist and that no invalid or unexpected categories are introduced due to sensor error, data corruption, or transmission failures.

Key Aspects of Category Completeness:

- **Balanced Category distributions:** If important categories are too scarce (e.g., few observations for severe rainfall occurrences), a dataset with complete labels may not be informative[85]. A sufficient frequency guarantees that machine learning algorithms may discover significant patterns across categories.
- **Category Encoding Consistency:** The data storage schemes for categorical variables must be consistent across the dataset. For instance: "SRAD," "srad," and "SRad" cannot coexist as distinct or difference values. Inconsistent encoding of binary scenarios must be avoided (e.g., utilizing both 0/1, True/False and Yes/No). Dependable feature engineering and analysis of models are ensured by consistent uniform category encoding [85].
- **No Missing Category:** This assesses if the dataset has all necessary or theoretically viable categories and if all categories are well represented. For instance, in a pollution dataset, all classes of pollution must be represented

and classified in a methodological order (like dangerous, moderate, unharmful).

Importance of Category Completeness:

Category completeness ensures that all required categorical values are present, valid, and complete in a dataset. It is important because it prevents misclassification and incorrect model learning, improves accuracy and fairness by avoiding biased categories, supports standardization and interoperability across systems and ensures reliable real-time processing in cloud and IoT applications[83]. Complete and standardized categories lead to more accurate, unbiased, and dependable machine learning outcomes.

3.4 Exploratory Data Analysis

Exploratory Data Analysis is a technique for analyzing data to highlight its properties, test hypotheses, and spot any abnormalities before beginning formal modeling. The information concealed in the rows and columns of data are better visualized, summarized, and comprehended [56]. EDA makes it possible to find underlying patterns, trends, and anomalies that are crucial for directing further phases of data preparation and completeness checks. This section describes the EDA methods and instruments employed in the study and talks about the conclusions drawn from the analysis.

Exploratory Data Analysis (EDA) is a critical component of this study, as it guides the necessary steps for understanding the statistical and structural characteristics of the Air Quality Index dataset before building predictive models. The purpose of EDA in this context is not to produce final results, but to clarify the analytical strategies used to uncover data behaviour, highlight potential quality issues, and determine the most suitable modelling approaches. As emphasised by Yang in [57], EDA forms the bridge between raw data preprocessing and model development, enabling the analyst to first understand patterns before attempting prediction.

Given that meteorological sensor data are often complex, non-linear, and subject to temporal dependencies, EDA becomes particularly important for capturing deeper relationships among variables. Weather datasets frequently present multicollinearity, outlier sensitivity, high-frequency fluctuations, and irregular variations caused by environmental processes or sensor instability. As noted by Chatfield in [58], time-dependent environmental measurements require especially careful exploration to avoid misleading statistical interpretations. Therefore, the EDA procedures in this study are designed to ensure conceptual clarity, diagnostic reliability, and methodological justification before moving to model training. Additionally, EDA supports the completeness assessment conducted in earlier sections by revealing structural gaps such as missing intervals, inconsistent timestamps, or attribute-level dropouts common in automated weather stations.

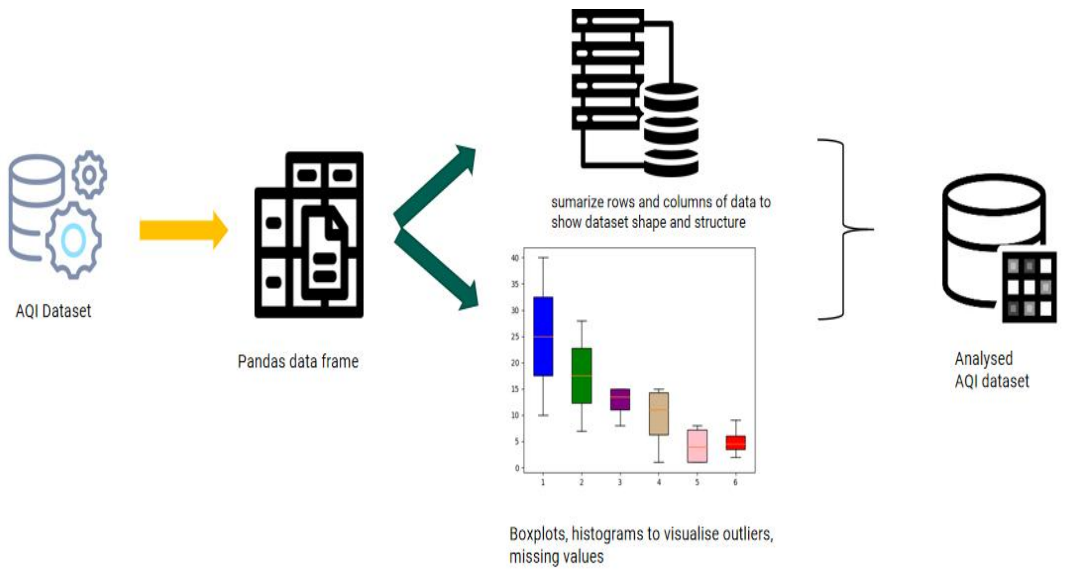

AQI Dataset
Pandas data frame
sumarize rows and columns of data to show dataset shape and structure
Boxplots, histograms to visualise outliers, missing values
Analysed AQI dataset

Figure 3.2: Exploratory Data Analysis (EDA)[57].

3.4.1 Techniques of EDA Used

The Exploratory Data Analysis (EDA) stage combines descriptive statistics and visual techniques to reveal the fundamental properties of the dataset. These analyses provide insights into the structure, variability, and relationships within the data, enabling informed decisions for subsequent modelling. The following procedures were applied [59]:

- **Statistical Summary Measures:** Core descriptive metrics including mean, median, standard deviation, and inter-quartile range were computed to describe central tendency, spread, and distributional shape. These statistics were useful for identifying incomplete values, possible anomalies, and overall variability across features.

- **Data Visualization Techniques:** Visual exploration supported the identification of distributional patterns, dependencies, and irregularities:

Box Plots:

Box plots are a fantastic way to depict the concentration of data graphically. It displays the outlier, skew, symmetry, and central tendency. The minimum, first quartile, median, third quartile, and maximum value are the five values that may be used to create it. To demonstrate how similar these numbers are to other data values, they are compared. Box plots summarize median values, quartiles, and extreme observations, offering a clear view of spread, symmetry, and anomalies. They provided an effective tool for detecting outliers and evaluating dispersion.

Scatter Plots:

A scatter plot is a graphical tool that represents the relationship between two variables using Cartesian coordinates. Each variable is assigned an axis, with one plotted along the horizontal (x-axis) and the other along the vertical (y-axis). The data points are displayed as individual dots, where the position of each point reflects the paired values of the two variables. This visualization helps to identify patterns, trends, and possible correlations within the dataset.
Scatter plots displayed pairwise relationships between variables using Cartesian coordinates across all features. These visualizations helped detect multicollinearity and guided decisions about feature selection to avoid redundant inputs.

Histograms:

Histograms illustrate the frequency distribution of continuous variables through rectangular bins, where width represents class interval and height reflects frequency density. These plots were used to assess skewness, kurtosis, and potential outliers in individual attributes.

- **Python Tools for EDA:** Several Python libraries facilitated data handling, statistical analysis, and visualization:

Pandas:

Pandas library was used for robust data cleaning, manipulation, and computation of summary statistics. Functions such as describe() were particularly useful for organizing and inspecting attribute-level characteristics.

Matplotlib:

In this research, matplotlib provided a flexible plotting framework for constructing graphical representations such as histograms and scatter plots, enabling detailed customization to highlight specific analytical observations.

Seaborn:

Built on top of Matplotlib, Seaborn simplified the creation of statistical plots, including box plots and heatmaps, with improved aesthetics that enhanced interpretability. Its high-level functions supported efficient exploration of trends, correlations, and outliers.

Together, these statistical and visualization approaches provided a comprehensive understanding of the dataset’s distributional properties, temporal behaviour, and structural consistency, forming a reliable foundation for subsequent feature engineering and predictive modelling.

3.4.2 Univariate Analysis of the Target Variable

The most basic type of data analysis that concentrates on looking at one variable at a time is univariate analysis. Without taking into account correlations with other variables, the objective is to summarize and comprehend the salient features of that particular variable [60]. Univariate analysis clarifies which value (average, median) is typical, what the standard deviation and variance of the values are, which distribution type (normal, skewed, or heavy-tailed) is it, and if there are noticeable outliers in a data. All these are helpful in understanding the behavior of a variable prior to modeling.

Why is Univariate Analysis Important?

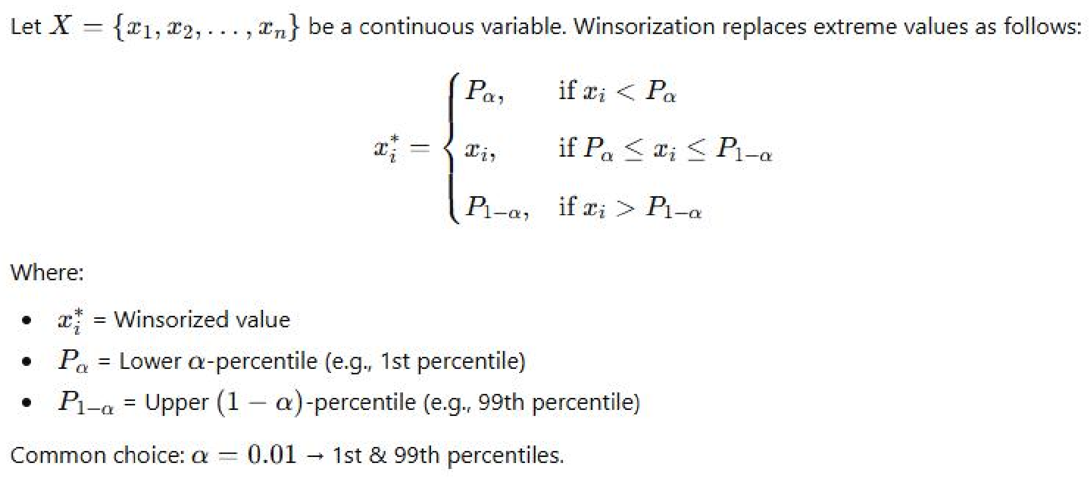
Because it gives researchers a fundamental knowledge of each variable separately, univariate analysis is crucial for identifying anomalies, outliers, or abnormal values that could skew statistical measurements or machine learning models. It assists in locating skewed distributions, gaps, or discrepancies that may require preprocessing, such scaling or adjustments in order to satisfy modeling algorithms' predictions. Univariate analysis guarantees that features are significant, well structured, and appropriate for predictive modeling by carefully describing individual variables. This enhances the accuracy, interpretability, and dependability of the model. Additionally, it makes it possible to make well-informed decisions during feature engineering and selection,laying a foundation for more intricate bivariate or multivariate studies[60].

The exploratory data analysis engaged in an extensive univariate assessment of the target variable, BarometricPressure. The barometric pressure readings were first inspected using histograms and kernel density estimators (KDEs) to visualize the general distribution shape and assess characteristics such as skewness, kurtosis, and possible multi-modal patterns. To further evaluate the degree of normality, Q–Q plots were generated and compared to a theoretical Gaussian distribution. While normality is not a requirement for most machine learning models, departures from

normal distributions are useful indicators of whether linear approaches may be adequate or whether more flexible nonlinear models are preferable.
Additionally, boxplots were used to assess the spread of barometric pressure values and highlight potential extreme observations or irregular variations. This univariate analysis aimed to characterize the statistical behaviour of the target variable and inform subsequent modelling decisions, including model selection, transformation needs, and performance metric suitability. The results of these visual and statistical analyses after implementation will be discussed in the later chapters.

3.4.3 Winsorization and Outlier Detection

Winsorization is a statistical transformation strategy that replaces extreme results (outliers) with certain percentile-based criteria instead of eliminating them in order to lessen their impact. The values are limited to upper and lower bounds rather than eliminating aberrant observations, which could include significant atmospheric fluctuations[61]. Mathematically, Winsorization is defined as follows:



Let $X = \{x_1, x_2, \ldots, x_n\}$ be a continuous variable. Winsorization replaces extreme values as follows:

$$x_i^* = \begin{cases} P_\alpha, & \text{if } x_i < P_\alpha \\ x_i, & \text{if } P_\alpha \leq x_i \leq P_{1-\alpha} \\ P_{1-\alpha}, & \text{if } x_i > P_{1-\alpha} \end{cases}$$

Where:

- x_i^* = Winsorized value
- P_α = Lower α-percentile (e.g., 1st percentile)
- $P_{1-\alpha}$ = Upper $(1-\alpha)$-percentile (e.g., 99th percentile)

Common choice: $\alpha = 0.01$ → 1st & 99th percentiles.

Identifying and managing outliers is an essential data-preparation step when working with meteorological measurements, as sensor-based weather data may be affected by mechanical faults, communication losses, or atmospheric noise. As emphasized in [62], extreme values can distort statistical and machine learning models if left untreated, yet their removal may lead to the loss of valuable information, particularly in environmental datasets.
Outlier detection in this study employed a combined approach, using both statistical criteria and visual inspection. Outliers were flagged using the z-score threshold method and the interquartile range (IQR) rule, while supplementary inspection through boxplots and time-series visualisations helped confirm anomalous patterns.

To avoid discarding potentially meaningful meteorological extremes, winsorization was implemented. Rather than removing extreme values, winsorization replaces them with less extreme threshold values, typically determined from upper and lower percentiles (e.g., the 1st and 99th percentiles).

Why is Outlier Detection Important?

Means, variances, correlations, and model coefficients can all be distorted by outliers, which can skew parameter estimates and lower prediction accuracy. By identifying and correcting these extreme values, the dataset is guaranteed to represent actual behavior and conform to the presumptions of several learning algorithms. By restricting extreme observations to predetermined percentile thresholds rather than eliminating them, winsorization, in particular, lessens the negative effects of outliers while retaining sample size and the general structure of the data. These methods increase model resilience, boost generalizability, and yield more dependable and comprehensible findings by lowering the leverage of unusual values while preserving significant information[61].

Winsorization was selected because meteorological events often display genuine extremes, such as rapid barometric pressure changes associated with storms or weather fronts. Such features should not be removed but controlled statistically to prevent undue influence on models. This chapter describes the methodological rationale; the effects and outcomes of winsorization will be evaluated and discussed in Chapter 4.

3.4.4 Pearson and Spearman Correlation Analysis

In machine learning, correlation is a statistical method used to find relationships between many variables. Correlation describes how two variables are connected by indicating both how strongly they are related and in what direction the relationship occurs. The size of the correlation value reflects the strength of the association, while the sign shows whether the variables move in the same direction (positive correlation) or in opposite directions (negative correlation). Different statistical techniques are used to measure correlation, but the two most widely applied in research and predictive modelling are Pearson’s correlation, which evaluates linear relationships, and Spearman’s correlation, which assesses monotonic relationships based on ranked data[62].

Pearson’s Correlation:

Pearson's correlation measures the linear relationship between two variables to determine the strength of their linear dependence[62]. It can be mathematically defined as:

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

Where,

r = Pearson Correlation Coefficient

x_i = x variable samples y_i = y variable sample

$\bar{x}$ = mean of values in x variable $\bar{y}$ =mean of values in y variable

Spearman’s Correlation:

Based on the rank of the data, the Spearman correlation quantifies a monotonic connection between two variables. [5,40,84] becomes [3,2,1] as an illustration of data rank determination in spearman correlation[62]. When dealing with data that contains outliers, Spearman correlation is frequently employed. The Spearman coefficient rs, often referred to as the rank coefficient, is the indicator used to calculate Spearman correlation and is provided by the following formula:

$$r_s = 1 - \frac{6\sum d^2}{n(n^2-1)}$$

v

The variable n in this calculation represents the data series' point count. The square of the rank difference between each point with coordinates (x,y) is represented by the variable, d.

Application of Pearson and Spearman Correlation:

Correlation analysis was carried out to evaluate the strength and direction of association between the meteorological variables, with the aim of guiding informed feature selection for predictive modelling. Both Pearson’s correlation coefficient and Spearman’s rank correlation coefficient were employed in this study. Pearson’s method quantifies the degree of linear association between two continuous variables and is most appropriate when the expected relationship is approximately linear. In contrast, Spearman’s correlation measures monotonic relationships using ranked values, making it more resistant to non-normal distributions, skewness, and outliers[62].

Correlation matrices were generated to visualize pairwise associations across all variables. These matrices are particularly useful for identifying highly correlated features, detecting redundant predictors, and highlighting variable clusters that may influence model stability. Furthermore, scatterplots with fitted trend lines were produced to visually examine whether relationships between predictors and the target variable (BarometricPressure) exhibit linear or monotonic behaviour. The choice of visual diagnostics ensures that decisions regarding feature inclusion, transformation, or regularization are grounded in the actual structure of the data rather than relying on assumptions.

3.4.5 Multicollinearity and Non-Linear Relationship Assessment

Multicollinearity refers to a statistical phenomenon in which two or more independent (predictor) variables in a regression model are highly correlated with each other. This means that the predictors provide overlapping or redundant information, making it difficult for the model to determine the individual effect of each variable on the outcome[63]. In regression, each predictor is expected to contribute unique information to explain the dependent variable. When predictors are strongly correlated, the model cannot distinguish their individual contributions because changes in one predictor are closely associated with changes in another. This causes instability and distortion in model estimation.

For example: Let’s assume the Air Temperature attribute is being predicted using humidity and rain which are themselves strongly related. The model struggles to isolate whether humidity or rain is driving changes in temperature, because they move in similar ways.

Detecting multicollinearity involves common techniques such as and not limited to variance inflation factors (VIFs), scatterplot matrix and examination of eigenvalues[64]. For the purpose of this study, the principle of variance inflation factor and scatterplots for visualization of variable relationships were employed in tackling multicollinearity.

Variance Inflation Factor (VIF)[63]:

General mathematical formula for multicollinearity is given as follows:

$$VIF = \frac{1}{1 - Rj2} j = 1,\ 2, \ldots, P - 1 \tag{1}$$

Where ***Rj2*** represents the coefficient of determination for the regression of the independent variables of ***jth*** on the remaining ones which are unadjusted.

Mutual information is one metric used by VIF to determine dependencies (both linear and non-linear) that are between variables. It measures the entropy **H(Y)** as follows[63]:

$$H(Y) = -p(y)\, logp(y) \tag{2}$$

When the value of ***X*** is known, the actual value of ***Y*** can be calculated by **H(Y|X)** as follows:

$$H(Y|X) = \sum p(x) \ * \ H(Y|X = x) \tag{3}$$

H(Y|X) also has the equivalence:

$$H(Y|X) = H(Y;X) - \ H(X) \tag{4}$$

Therefore, mutual information between **X** and **Y** can be measured as follows:

$$I(X;\, Y) = H(Y) - H(Y|X) \tag{5}$$

Given the case where multivariate factors are present, mutual information can be estimated by measuring the mutual information ratio (MIR) between **X** and **Y** as follows:

$$MIR == \frac{I(X;\, Y)}{H(Y)} \tag{6}$$

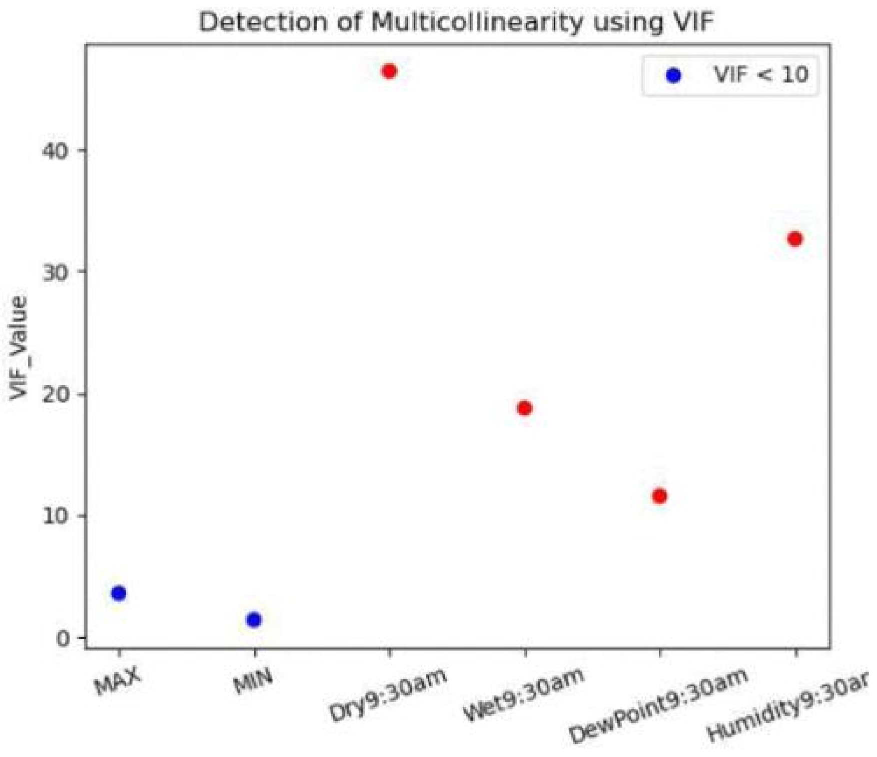


Figure 3.3: Detection of Multicollinearity using Variance Inflation Factor[63].

Application of Multicollinearity using VIF :

The AQI dataset was assessed for potential non-linear relationships, which are common in atmospheric systems due to complex interactions between variables such as humidity, solar radiation, pressure gradients, and wind behaviour. To evaluate non-linearity systematically, Mutual Information (MI) was used to rank the importance of numerical features in predicting the target variable, Variance Inflation Factor (VIF) scores were evaluated to measure how much the variance of each feature is inflated due to multicollinearity. Furthermore, smoothed scatterplots, residual pattern inspection, and shape diagnostics were applied. These diagnostics help determine whether a linear modelling strategy is sufficient or whether more flexible machine learning models such as Random Forest, XGBoost, or neural networks, are required to capture the underlying structure of the data.

Why is Multicollinearity Calculation Important:

When predictors exhibit strong interdependence, the model finds it hard to distinguish their individual effects, resulting in unreliable coefficient estimates and inflated standard errors[63]. This instability can lead to misleading conclusions regarding variable significance and undermine the scientific interpretation of atmospheric relationships. Furthermore, high multicollinearity may reduce the generalization performance of a model, as redundant features increase the likelihood of over-fitting. Understanding the extent of multicollinearity therefore supports more informed feature engineering decisions, such as removing or combining correlated predictors, and guides the selection of appropriate modelling techniques[64]. While linear models are particularly vulnerable to multicollinearity, regularized and ensemble-based machine learning models can mitigate its effects, making its assessment vital for coherent and robust model development [65].

3.5 Model Training and Comparison

Model training and comparison constitute the core analytical stage of this research, during which the cleaned and pre-processed AQI dataset is utilized to develop predictive models capable of estimating atmospheric pressure from a range of environmental variables. The primary aim of the modelling framework is to systematically evaluate the effectiveness of several machine learning algorithms, specifically Decision Trees, Random Forests, and XGBoost, in capturing the complex, nonlinear, and interactive patterns characteristic of meteorological systems. These algorithms were selected due to their strong empirical performance in environmental prediction tasks as well as their capacity to provide both high predictive accuracy and interpretable importance metrics[66],[67],[68]. All modelling processes were implemented in Python using the scikit-learn and XGBoost libraries, ensuring reproducibility and alignment with widely adopted machine learning standards.

Random Forest Regressor (RFR):

The Random Forest Regressor (RFR) model was one of the three chosen modeling tool in this study because of its resilience in managing high-dimensional datasets and its capacity to capture intricate, non-linear interactions using an ensemble approach [72]. There are several independent and uncorrelated decision trees F = {t1, t2,..., tT } that make up the Random Forest model[73]. The bagging approach combines random sampling with replacement, also called bootstrapping as well as aggregation to provide predictions with reduced variance and increased accuracy. This approach is then used to train these independent and uncorrelated trees. In order to obtain a balanced generalization that reduces over fitting, this ensemble model makes use of the randomness introduced into each tree.

The training set defined as:

$$S = \{(X_m, Y_m)\}_{m=1}^{M}$$

Where ***X ⊂ RD*** is the feature space, consisting of attributes such as humidity, rain, SRAD and other meteorological variables and ***Y ⊂ RD*** represents the continuous target variable, Barometric Pressure.

The training produces a final arithmetic mean of output as follows:

$$\hat{y} = \frac{1}{T} \sum_{t=1}^{T} t(X)$$

The bootstrap sample ***St*** is used to train each decision tree in the forest and a final aggregated prediction, which is more reliable than predictions from a single tree, is produced during prediction by averaging the outputs of the different trees.

Extreme Gradient Boosting (XGBoost):

The Extreme Gradient Boosting (XGBoost) model was also evaluated as a predictive algorithm due to its capability to optimize computational speed, handle large datasets, and efficiently capture non-linear dependencies using a boosting-based ensemble framework. XGBoost, originally proposed by Chen and Guestrin in [74], uses a scalable implementation of gradient boosting where multiple weak learners (typically regression trees) are trained sequentially, each correcting the errors of its predecessors.

Considering again the training dataset in RFR model above and their representatives, XGBoost creates a function ensemble where each ***fk*** belongs to a space of regression trees ***F*** in the equation below:

$$\hat{y}(x) = \sum_{k=1}^{K} f_k(x)$$

The objective function minimized during training consists of a loss term ***L(.)*** and a regularization term ***Ω(fk)***, given by:

$$\text{Obj} = \sum_{m=1}^{M} L(y_m, \hat{y}_m) + \sum_{k=1}^{K} \Omega(f_k)$$

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \|w\|^2$$

where ***T*** is the number of leaf nodes and ***w*** are the leaf weights. The inclusion of the L2 penalty term ***λ*** controls model complexity and reduces over-fitting, a critical benefit when working with environmental data that may contain noise, extreme spikes, or multicollinearity among features[74].

The boosting process updates trees by fitting residuals in the negative gradient direction. At iteration ***t***, an additional tree ***ft(x)*** is added as:

$$\hat{y}^{(t)} = \hat{y}^{(t-1)} + \eta f_t(x)$$

where ***η*** is the learning rate controlling contribution of new trees. Unlike Random Forests (which emphasize variance reduction), XGBoost improves accuracy through additive error correction, shrinkage, regularization, and column sampling, making it robust for high-dimensional meteorological prediction tasks.

Decision Tree Regressor (DTR):

Decision Tree Regression was also studied as a baseline model for interpreting structural dependencies within meteorological data. A Decision Tree splits the input feature space recursively into homogeneous sub-regions using threshold rules. These rules maximize reduction in impurity measures, commonly variance reduction for regression tasks [76].

Given again the training dataset in RFR above, the model partitions the feature space by selecting features and thresholds that minimize the sum of squared errors (SSE) at each node:

$$\mathrm{SSE} = \sum_{m \in R_1(j,s)} (y_m - \bar{y}_{R_1})^2 + \sum_{m \in R_2(j,s)} (y_m - \bar{y}_{R_2})^2$$

where ***$R_1(j,s)$*** and ***$R_2(j,s)$*** are the resulting regions after splitting feature ***j*** at threshold ***s***, and ***$y^{-}R_1, y^{-}R_2$*** are mean responses in each region. Leaves of the tree assign predictions equal to the mean value of the training samples within the terminal region:

$$\hat{y}(x) = \bar{y}_{R(x)}$$

Decision Trees have strong benefits such as interpretability, low computational cost, and no strict assumptions about linearity or distributional shape. However, they are highly sensitive to noisy sensor readings, missingness, and correlated atmospheric attributes[75]. This instability motivates the use of ensemble variants such as Random Forest and Gradient Boosting (including XGBoost), which reduce error and increase generalization by combining multiple trees rather than relying on a single estimator.

3.5.1 Data Splitting and Training Strategies

The modelling workflow begins by dividing the complete dataset into separate training and testing subsets. This step is essential for ensuring that model evaluation is conducted on unseen data, allowing performance metrics to reflect true predictive capability rather than memorization. The **train_test_split** function from scikit-learn was used to allocate **80%** of the data to the training set and **20%** to the testing set. Importantly, the split was performed only after all missing values had been removed to prevent information leakage. A randomized split was considered appropriate because the dataset had already been shuffled and did not contain strong temporal dependencies (e.g., hourly sequences).

Following the initial split, each model was further evaluated using five-fold cross-validation implemented via scikit-learn’s KFold procedure with (shuffle=True). Cross-validation provides a robust assessment by repeatedly training the model on different subsets of the data and evaluating it on corresponding validation folds[69]. For each fold, the model was re-initialized and fitted from scratch using the fold’s training portion, then evaluated on the validation portion. Employing shuffled folds reduces structural bias, and given that the models are non-parametric and largely insensitive to data ordering, this approach supports reliable error estimation[70].

Three machine learning algorithms were examined using this multi-stage validation procedure: the Decision Tree Regressor, Random Forest Regressor, and XGBoost Regressor. This layered training protocol is consistent with contemporary forecasting research, which emphasizes the importance of rigorous validation to mitigate over fitting in complex predictive models[71]. Together, the training–testing split and cross-validation framework ensure that the models are assessed for stability, consistency, and generalization performance in a systematic and methodologically manner.

3.5.2 Model Evaluation Techniques

To rigorously evaluate the performance of the three selected regression models—Random Forest Regressor (RFR), Decision Tree Regressor (DTR), and XGBoost Algorithm, two standard error metrics were employed: Root Mean Squared Error (RMSE), and R-Squared (R^2 score) [72]. These metrics were chosen to ensure a complete and comprehensive assessment of predictive accuracy and stability across both the raw and Improved AQI datasets. For the RFR model, these metrics help highlight its robustness and ability to reduce variance through the aggregation of multiple uncorrelated trees, leading to improved generalization[73]. In contrast, the DTR model, while capable of capturing nonlinear relationships, is more prone to over-fitting; therefore, using these metrics provides valuable insight into its sensitivity to noise and data variability. Meanwhile, XGBoost combines boosting with regularization, iteratively minimizing residual errors, and the selected metrics play a key role in quantifying its enhanced accuracy and ability to prevent over-fitting through controlled learning [74]. Overall, R-Squared and RMSE effectively capture the error distribution and model reliability, facilitating a fair performance comparison among all three regressors on fault prediction tasks[72].

1. Root Mean Squared Error (RMSE)

RMSE is defined as the square root of Mean Squared Error (MSE), which returns the prediction error to the same units as the target variable, making it more interpretable than MSE. By taking the square root of the mean squared errors, RMSE preserves the advantages of MSE, particularly its sensitivity to larger deviations, while ensuring the metric remains directly comparable to the scale of the observed data [72]. Practically, RMSE offers an overall measure of model accuracy that balances the interpretability of Mean Absolute Error (MAE) with the error sensitivity of MSE. RMSE reflects both the variance and the bias present in model predictions providing an understandable indicator of the predictive performance of the model. For this reason, RMSE is widely used in applications where understanding the magnitude of real-world error is essential for evaluating model reliability.
RMSE is mathematically represented as follows:

$$\mathrm{RMSE} = \sqrt{\mathrm{MSE}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Importance:
RMSE is sensitive to large errors which are common in real world complex datasets (such as AQI dataset), across domains while balancing accuracy in predictions.

2. **R-Squared (R^2 score)**
Formula:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Where:

- y_i = actual values
- $\hat{y}_i$ = predicted values from the model
- $\bar{y}$ = mean of actual target values
- n = number of observations

R-squared (R^2), can also be called the coefficient of determination or score, measures the proportion of variance in the target variable that can be explained by the model[75]. It quantifies how well the model fits the observed data by comparing the model's predictive performance against a baseline model that simply predicts the mean of the target variable. R^2 values range between 0 and 1, where a value closer to 1 signifies a strong explanatory power, indicating that a large portion of the data variability is captured by the model. Conversely, lower values of R^2 indicate weaker explanatory capability and suggest that the model fails to capture important underlying patterns. Unlike error-based metrics such as MAE, MSE, and RMSE, R^2 provides a relative measure of model fitness evaluation, making it especially useful for comparing different models on the same dataset.

Importance:
R-squared score measures the observation of goodness of fit (model fitness evaluation) to examine how well the dataset fits the model and this is important in this research given the comparison of the three models on the same AQI dataset.

3.5.3 Feature and Permutation Importances

In order to evaluate which predictors were most important for the model prediction, feature importance and permutation importance of the predictors for each of the models were calculated using scikit-learn. This enhances accurate interpretation of the model training processes. First, cross-validation and selection of the best hyperparameters for each model occurred, followed by retraining of all models using the entire training dataset. Finally, two complementary approaches for feature importance computation were applied: the model based importance (which is feature importance) and the permutation importance. The feature and permutation scores of each model RFR, DTR and XGBoost will be discussed in detail in the later chapter.

Feature Importance

Feature importance is a method for measuring each predictor's contribution to the prediction performance of a machine learning model[77]. By concentrating on the most useful variables, it helps researchers comprehend, interpret, and maybe simplify models by revealing which aspects have the most impact on the model's predictions. This is especially useful for lowering dimensionality without compromising performance, directing feature selection, and enhancing model interpretability.

Permutation Importance

In contrast to model-dependent importance metrics, permutation importance does not rely on the internal structure of the model, but rather measures how much the performance of the model decreases by randomly shuffling either of the features. This technique is to offer a less biased and model-agnostic estimate of the true contribution of each predictor and ensures a complete and balanced interpretability evaluation by using both the internal importance and external importance framework[78].

3.5.4 Feature Aggregation and Ranking

After calculating the six importance measures, three from each of the models (Decision Tree, Random Forest, XGBoost) and two from each of the interpretability methods (model-based and permutation-based) , the next methodological step was to combine those measures into a single ranking. The model averaging process guarantees that the final feature importance will not show high variance with a particular model specification. A DataFrame was created where each column type was one of the model’s feature and permutation importance list. Also, normalizing the importance scores enhances the comparison of the importance of inputs across the three different models as they produce scores at different scales. Finally, an average importance score was calculated for each feature across all six metrics and the most frequent associations sorted in descending order of ranking. The ensemble-based ranking approach aggregates information of the feature and permutation importance from multiple models to yield a more stable estimate of feature importance[79].

4 Implementation and Design

This chapter discusses the architectural and methodological implementation of the proposed system. It presents the detailed overall design principles, the rationale behind key technical decisions, and the processes used to translate the conceptual model into a functional solution. The chapter outlines the system architecture, component interactions, and the technologies employed, followed by the implementation workflow and integration strategy. Connecting the ideas in previous chapters together, these sections provide a clear account of how the proposed approach was translated into a working system to meet the research objectives, ensuring scalability and reproducibility.

4.1 Technological Introduction

The implementation of the proposed data completeness analysis framework utilizes different robust technological tools including python programming language and it's powerful libraries. These tools were effectively used at every step of the implementation to address the challenges of data completeness analysis and improvement in predictive modeling. This section explains all technologies employed in this research.

4.1.1 Python Language and Used Libraries

The python language was developed by Guido van Rossum and released the first time in 1991 as a simple, readable programming language designed to support rapid development[86]. Over time, its clean syntax, extensive standard library, and strong community support helped it grow into one of the most widely used languages in scientific computing, data analysis, and artificial intelligence[88]. The release of Python 3 in 2008 modernized the language and expanded its capabilities, enabling the rich ecosystem of libraries such as NumPy, Pandas, and TensorFlow, that define much of today's data-driven research [87].

Python is the main language used in this implementation because of it's flexibility, readability, and large library ecosystem. It is widely accepted across different industries due to it's scalability and efficiency. The use of python programming language in this research ensures robustness, maintainabilty and versatility.

Key Libraries Used

The python language is supported by a wide range of powerful libraries. The following libraries were utilized extensively for the implementation of the proposed framework:

- **Pandas:** Pandas library is a data structure for manipulating and analyzing data. It uses both one dimensional (series) and two-dimensional arrays (dataframe) in handling structured data efficiently, making it a foundational tool in data science. It was efficiently used to load and perform initial data cleaning on raw datasets, give overview of the findings of Exploratory Data Analysis, ensure the application of all completeness checks to improve the quality of dataset before model training.

- **Numpy:** Numpy enables efficient numerical operations and thus, form the computational backbone on which other libraries such as pandas, scikit-learn, pytorch etc, depend on. Numpy was used together with pandas for mathematical computations during data completeness checks.

- **Scikit-learn:** Scikit-learn provides an efficient framework for implementing a wide range of machine learning algorithms including regression. It was used to build and train both the Random Forest Regressor (RFR) and Decision Tree Regressor (DTR) models, evaluating their metrics using Root Mean Squared Error (RMSE) and R-squared score to determine the performance of the models.

- **Xgboost:** Xgboost is another machine learning library employed to build and train the Extreme Gradient Regressor (XGBoost Regressor), applying the RMSE and R-squared score evaluation metrics for proper comparison between itself, Random Forest Regressor and Decision Tree Regressor.

- **Matplotlib and Seaborn:** For data anomalies detection, these libraries were used to analyze datasets using boxplots, charts, scatterplots and histograms.

- **Scipy.Stats:** This library was used to identify data points that are disproportional and can affect model's estimation, apply winsorization to identify outliers and in Exploratory Data Analysis. It was used along side with **statsmodels.stats.outliers_influence** library.

- **Pathlib:** This is a modern python library which replaced **os.path** and handles filesystem paths, offering a more readable and platform-independent design necessary for uploading raw datasets and downloading processed dataset.

- **Streamlit:** The streamlit library was employed for the development of an interactive dashboard. visualization of data transformations during processing and Exploratory Data Analysis, as well as model training results. Its API allows seamless integration with Python data structures, machine learning models, and visualization libraries, enabling rapid prototyping and real-time interaction with analytical workflows.

4.1.2 Implementation Development Environments

The Implementation environments employed in the process of the data completeness analysis and improvement workflow were mainly Jupyter Notebook, Google Colab and Visual studio Code.

Jupyter Notebook

A key tool for the flexible and continuous creation of processes for the proposed data completeness analysis and data preparation was Jupyter Notebook and Google Colab. They enable developers to repeatedly evaluate multiple preprocessing approaches, including managing missing values, identifying anomalies, and employing The Cross-Industry Standard Process for Data Mining (CRISP-DM). These environments are suited for collaboration and real-time progress implementation. By blending Python code and Markdown text, Jupyter Notebook delivered a clear track of all preparation processes, making the procedures replicable and comprehensive. Libraries such as Matplotlib, Seaborn, and Pandas plotting functions were incorporated into the notebook to view data trends, plot graphs, view correlation heatmaps, and expose outlier indicators. Every stage of the completeness analysis process was verified in actual time to guarantee that techniques like data preprocessing, Exploratory Data Analysis (EDA), model training and regression, and CRISP-DM application yielded the desired outcomes.

Google Colab

In this research, Google Colab functions as a cloud-based computational environment that facilitates the development, training, and evaluation of machine learning models. Its hosted Jupyter Notebook interface provides access to GPU and TPU acceleration, enabling efficient execution of computationally intensive experiments without the need for local high-performance hardware. Google Colab's seamless integration with Google Drive supports collaborative work, version tracking, and organized storage of datasets and experimental outputs. Additionally, the preconfigured Python ecosystem reduces setup overhead, allowing the research to focus on experimentation and analysis rather than environment management. Overall, Google Colab enhances reproducibility, scalability, and accessibility throughout the implementation process.

Visual Studio Code

In this research, Visual Studio Code serves as the primary development environment for implementing, testing, and managing the project's software components. Its lightweight architecture, combined with an extensive ecosystem of extensions, supports efficient coding in Python, seamless integration with version control (Git), and interactive data exploration through Jupyter Notebook support. Visual Studio Code's debugging tools, environment management features, and integrated terminals streamline the development workflow, enabling consistent execution of experiments, rapid iteration of models, and effective organization of project files. As a result, the environment contributes significantly to the reproducibility, maintainability, and overall efficiency of the research implementation.

4.1.3 Relevance of TUC SmartCityCloud Infrastructure

The TUC SmartCityCloud Infrastructure provided the foundation for the integration of this research with its robust cloud services and database management. It made available, a testing and deployment environment for the proposed data completeness improvement and analysis workflow.

TUC SmartCityCloud Cloud Services

The SmartCityCloud platform functions as an advanced complex data storage and analytic environment designed to manage and process real-time sensor data thereby improving data quality. It is well suited for applications requiring structured data collection and preprocessing, completeness analysis workflows, high reliability validation, and easy integration with different external tools.
The following features of SmartCityCloud fueled the implementation of data completeness analysis:

- **Structured Data Collection:** Systematically collects structured time-series data in a categorized manner according to their timestamps and label them correctly. Sensor data are critically important when monitoring environmental behaviour for a given period of time. As such, data monitoring becomes easier when data is structurally collected to avoid data incompleteness.

- **Data Safety:** Data is always safe within SmartCityCloud given the regular synchronization of locally stored data to the cloud. This minimizes data loss which is a core aspect of data completeness check in overall data quality.

- **Easy Integration:** SmartCityCloud allows easy and problem-free integration with external tools for custom enhancements. Different visualization libraries like streamlit can be integrated for better dashboard representation as well as scripts with preferred model training capabilities and comparisons.

TUC SmartCityCloud Database

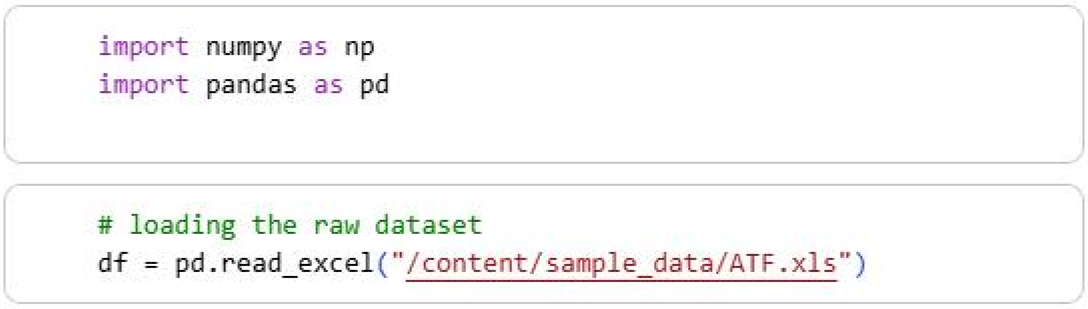
The TUC SmartCityCloud runs on MariaDB and MySQL Server which provides a reliable storage solution with well-defined schemas and relational databases for the proposed system. This is essential in storing both raw, improved and processed dataset. Upon launching the cloud platform, the database script initializes the database schema to set up for data storage, user logins and structured data manipulation involved in the data completeness workflow. Also, the database collaborates with tools from the web server for a better integration of the frontend visualization and representation. In general, the SmartCityCloud database supported the complete data preprocessing and model training checks of the proposed system.

4.2 Data Completeness Implementation

Having discussed all technological tools and environments employed for the implementation of data completeness analysis and improvement in the last section, this section aims to explain the methodological processes followed systematically to achieve the objectives of this research starting from data processing to model training. Data cleaning and pre-processing is an important step involved in the implementation pipeline which ensures that the incoming raw data is converted into a reliable, consistent, and analysis-friendly dataset. The subsequent subsections discuss how missing values, incorrect sensors placeholders, out of range values, temporal inconsistencies, duplicated observations and feature engineering processes were handled. Every procedure can be replicated exactly, as show in the code and outputs.

4.2.1 Loading the Raw Dataset

The implementation starts with importing the basic scientific computing libraries in python that are required for data pre-processing. This research uses NumPy for numerical operations and Pandas for DataFrame manipulation. Following the library loading process, the original raw dataset was loaded from the ***ATF.xls*** excel file using the `read_excel()` pandas function. Pandas automatically analyzed the structure of the columns and evaluated the data types of each column.

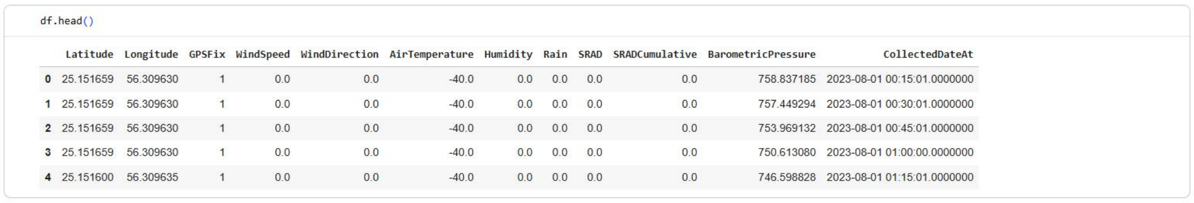


```
import numpy as np
import pandas as pd
```

```
# loading the raw dataset
df = pd.read_excel("/content/sample_data/ATF.xls")
```

Figure 4.1: Loading the dataset.

Successfully performing the import command, the `head()` function is called on the loaded data frame to display the first 5 rows and all columns of the dataset by default.



```
df.head()
```

	Latitude	Longitude	GPSFix	WindSpeed	WindDirection	AirTemperature	Humidity	Rain	SRAD	SRADCumulative	BarometricPressure	CollectedDateAt
0	25.151659	56.309630	1	0.0	0.0	-40.0	0.0	0.0	0.0	0.0	758.837185	2023-08-01 00:15:01.0000000
1	25.151659	56.309630	1	0.0	0.0	-40.0	0.0	0.0	0.0	0.0	757.449294	2023-08-01 00:30:01.0000000
2	25.151659	56.309630	1	0.0	0.0	-40.0	0.0	0.0	0.0	0.0	753.969132	2023-08-01 00:45:01.0000000
3	25.151659	56.309630	1	0.0	0.0	-40.0	0.0	0.0	0.0	0.0	750.613080	2023-08-01 01:00:00.0000000
4	25.151600	56.309635	1	0.0	0.0	-40.0	0.0	0.0	0.0	0.0	746.598828	2023-08-01 01:15:01.0000000

Figure 4.2: Displaying the first five rows of the dataset.

The print out of `df.head()` and the output as shown in the figure 4.2 above displays the first five observations that check for the ordering of the columns and the correctness of the initial values. This provides an early overview of the structure of the dataset and a check to confirm that dataset was properly read, loaded and presented as expected.

The head of the ***ATF.xls*** dataset contains the following 12 columns:

- **Latitude, Longitude:** Geospatial coordinates of the measurement point
- **GPSFix:** GPS positional fix indicator

- **WindSpeed and WindDirection:** Wind characteristics
- **AirTemperature and Humidity:** Atmospheric conditions
- **Rain, SRAD, SRADCumulative:** Precipitation and radiation metrics
- **BarometricPressure:** Target variable of interest
- **CollectedDateAt:** Timestamp of each observation

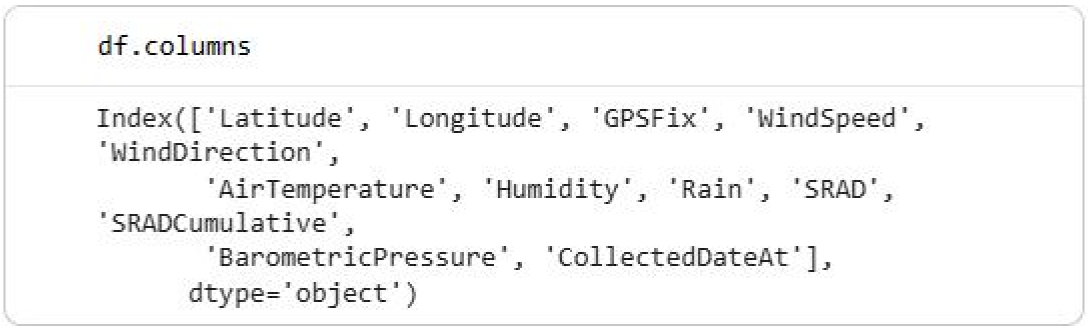
The Initial values of these rows indicate some invalid floating-point values for environmental measurements and timestamps formatted as Pandas object dtype, which will under processing as discussed in the later subsections of this implementation.

4.2.2 Missing Values Detection

After loading the dataset, an exploratory step was followed to verify the structure, completeness, and quality of the imported dataset in order to produce a transparent dataset for modeling using key pandas completeness functions.

df.columns Function

The `df.columns` function returns the 12 columns of the ***ATF.xls*** dataset which are aligned to environmental monitoring schema. This shows that the sensor export structure was preserved without field loss during data ingestion and all column labels correctly mapped.



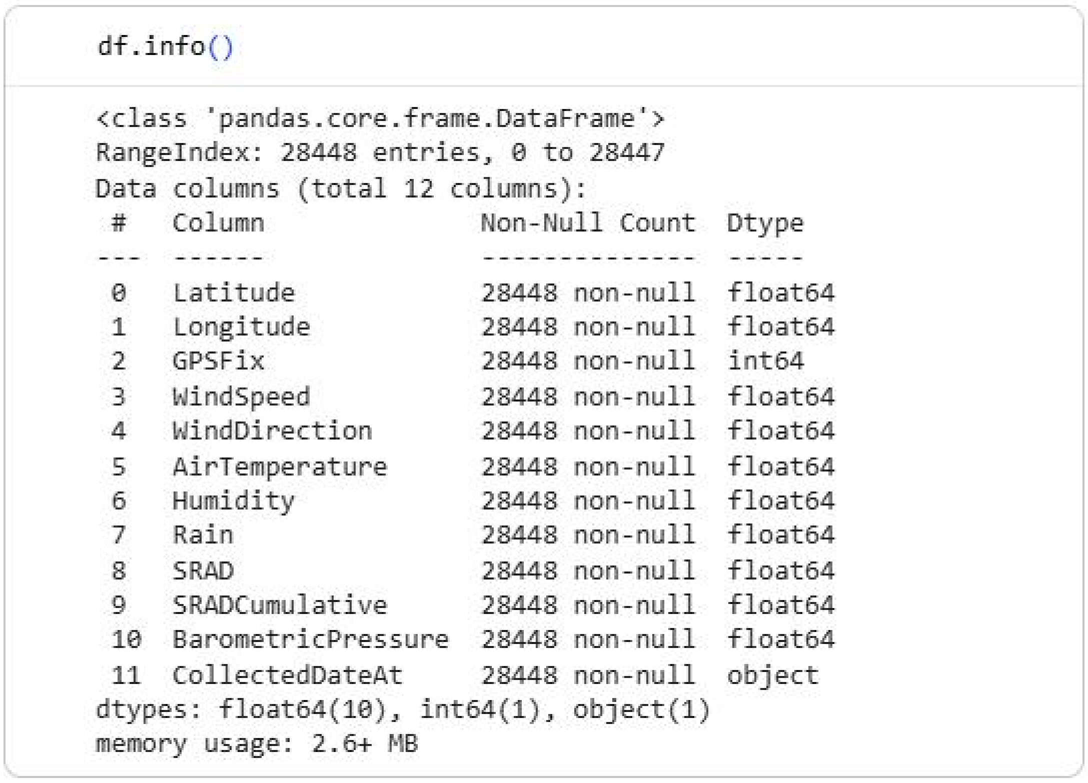
```
df.columns
```

```
Index(['Latitude', 'Longitude', 'GPSFix', 'WindSpeed',
'WindDirection',
       'AirTemperature', 'Humidity', 'Rain', 'SRAD',
'SRADCumulative',
       'BarometricPressure', 'CollectedDateAt'],
      dtype='object')
```

Figure 4.3: df.columns Function.

df.info() Function

The `df.info()` function was used to extract the data types of columns, their range index, number of non-null values and the memory usage of the dataset.



```
df.info()
```

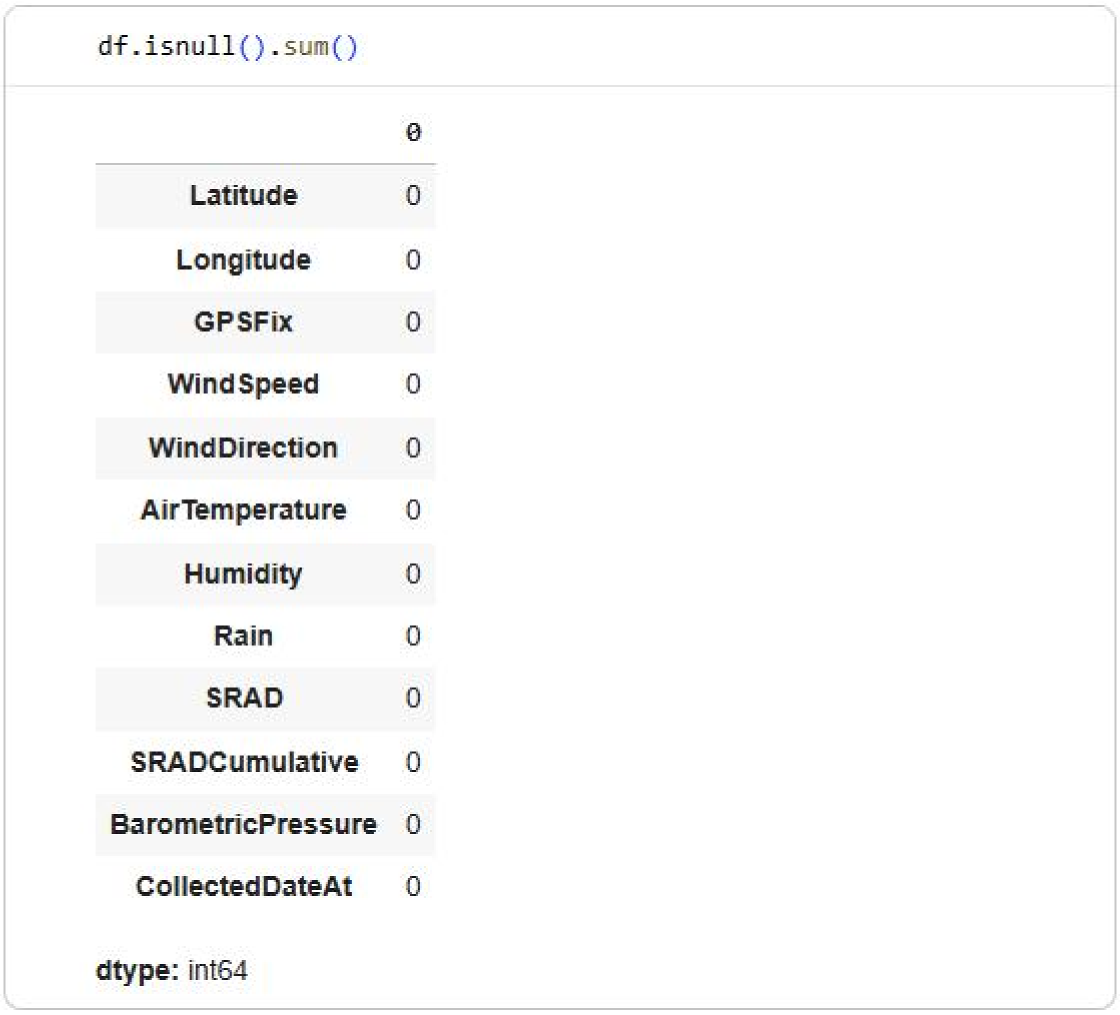
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28448 entries, 0 to 28447
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Latitude            28448 non-null  float64
 1   Longitude           28448 non-null  float64
 2   GPSFix              28448 non-null  int64  
 3   WindSpeed           28448 non-null  float64
 4   WindDirection       28448 non-null  float64
 5   AirTemperature      28448 non-null  float64
 6   Humidity            28448 non-null  float64
 7   Rain                28448 non-null  float64
 8   SRAD                28448 non-null  float64
 9   SRADCumulative      28448 non-null  float64
 10  BarometricPressure  28448 non-null  float64
 11  CollectedDateAt     28448 non-null  object 
dtypes: float64(10), int64(1), object(1)
memory usage: 2.6+ MB
```

Figure 4.4: df.info() Function.

The output of df.info() in figure 4.4 above shows that all numerical variables are stored as data type float64 except GPSFix, which is stored as data type int64. The field CollectedDateAt (a timestamp) is stored with the object data type which will be converted to a standard pandas datetime object for consistency. Every column has 28,448 non-null entries, suggesting that there are no missing values in any field with a memory consumption of about 2.6 MB.

df.isnull().sum() Function

The df.isnull().sum() function specifically returns the total sum of missing values in each column. The ***ATF.xls*** dataset showed no missing values in all columns as shown in figure 4.5 below:



```
df.isnull().sum()
```

```
                     0
          Latitude   0
         Longitude   0
            GPSFix   0
         WindSpeed   0
     WindDirection   0
    AirTemperature   0
          Humidity   0
              Rain   0
              SRAD   0
     SRADCumulative  0
BarometricPressure   0
    CollectedDateAt  0

dtype: int64
```

Figure 4.5: df.isnull().sum() Function.

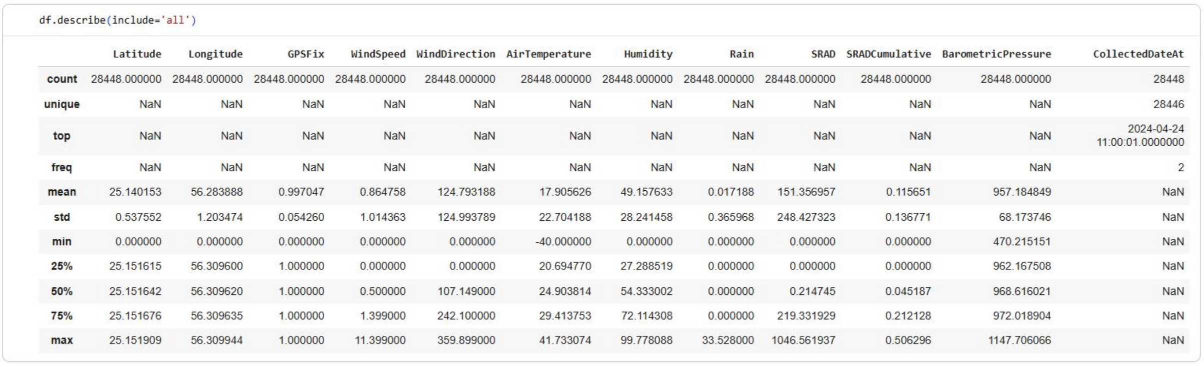
df.describe() Function

Although no formal missing values exist, placeholder values still represent invalid sensor readings, necessitating further cleaning procedures hence the need for descriptive statistics for range and pattern inspection generation using the df.describe(include=’all’) function.

The describe() function normally summarizes only columns with numeric values while include=’all’ tells pandas to include all columns irrespective of their data type; numeric, categorical, boolean, datetime, etc. The function provides a comprehensive overview of every column in a dataset, to understand data quality, distributions, and potential issues before modeling using the following statistics:

- **Count:** Number of non-missing (non-NaN) values.
- **Unique:** How many distinct entries exist in the column.

- **Top:** The most frequent (mode) value in the column.
- **Freq:** How many times the top value appears.
- **Mean:** The arithmetic average of the values.
- **Std:** Standard deviation, that is, how spread out the values are around the mean.
- **Min:** Minimum value in the column.
- **25%:** 25th percentile (lower quartile) means 25% of the values are below this number.
- **50% (median):** 50th percentile (middle value) indicates that half of the values are above and half are below this number.
- **75%:** 75th percentile (upper quartile) means 75% of the values are below this number.
- **Max:** Maximum value in the column.



```
df.describe(include='all')
```

	Latitude	Longitude	GPSFix	WindSpeed	WindDirection	AirTemperature	Humidity	Rain	SRAD	SRADCumulative	BarometricPressure	CollectedDateAt
count	28448.000000	28448.000000	28448.000000	28448.000000	28448.000000	28448.000000	28448.000000	28448.000000	28448.000000	28448.000000	28448.000000	28448
unique	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	28446
top	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2024-04-24 11:00:01.0000000
freq	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2
mean	25.140153	56.283888	0.997047	0.864758	124.793188	17.905626	49.157633	0.017188	151.356957	0.115651	957.184849	NaN
std	0.537552	1.203474	0.054260	1.014363	124.993789	22.704188	28.241458	0.365968	248.427323	0.136771	68.173746	NaN
min	0.000000	0.000000	0.000000	0.000000	0.000000	-40.000000	0.000000	0.000000	0.000000	0.000000	470.215151	NaN
25%	25.151615	56.309600	1.000000	0.000000	0.000000	20.694770	27.288519	0.000000	0.000000	0.000000	962.167508	NaN
50%	25.151642	56.309620	1.000000	0.500000	107.149000	24.903814	54.333002	0.000000	0.214745	0.045187	968.616021	NaN
75%	25.151676	56.309635	1.000000	1.399000	242.100000	29.413753	72.114308	0.000000	219.331929	0.212128	972.018904	NaN
max	25.151909	56.309944	1.000000	11.399000	359.899000	41.733074	99.778088	33.528000	1046.561937	0.506296	1147.706066	NaN

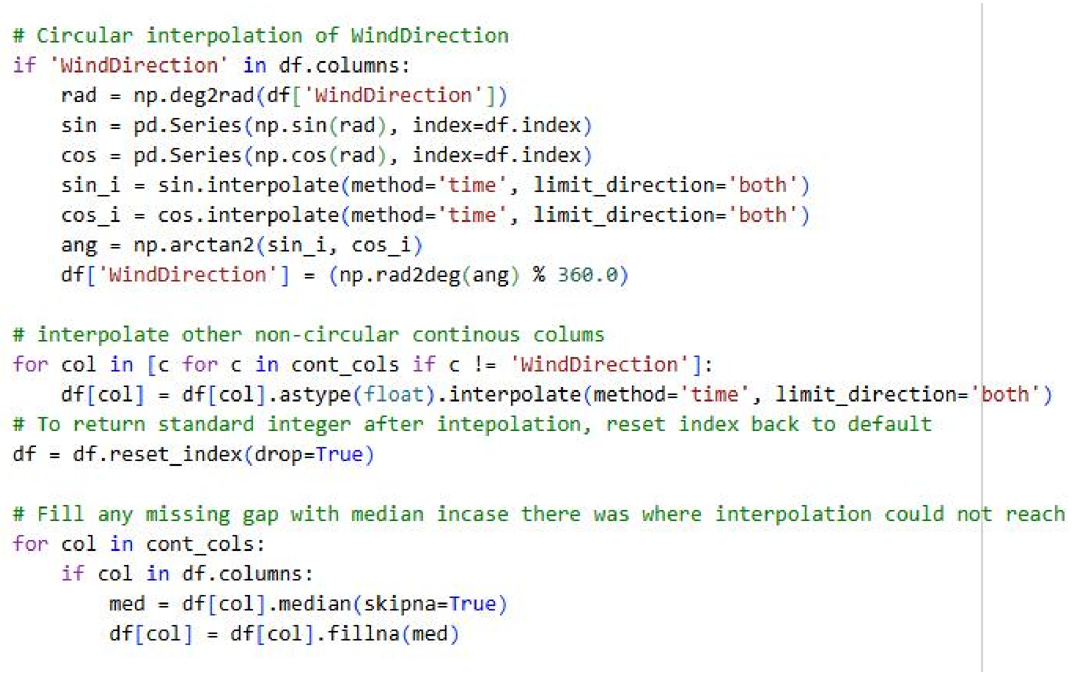
Figure 4.6: df.describe(include='all') Function.

From figure 4.6 above, it can be seen which statistics were not applied to different columns as the values were filled with NaN. The table produced contains the mean, standard deviation, minimum value, quartiles and maximum value of all numeric columns as well as unique, top and frequency values for datetime columns.

4.2.3 Data Accuracy

Data accuracy was achieved through interpolation and standardization of the continuous variables present in the ***ATF.xls*** dataset. AirTemperature, WindSpeed, and Humidity were interpolated over the time series to fill in the gaps. For WindDirection, trigonometric interpolation was applied using sine and cosine functions to resolve the circular nature of the variable so that the transition over

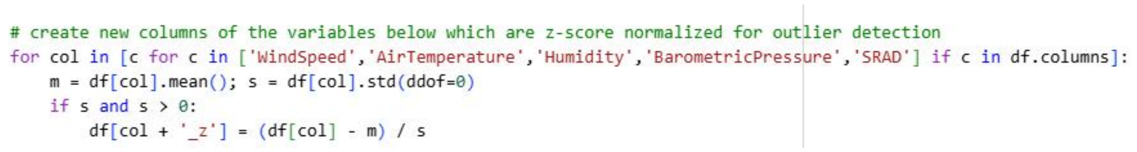
0°/360° is properly handled. Then, missing values were replaced by the median value to limit bias and outliers.



```python
# Circular interpolation of WindDirection
if 'WindDirection' in df.columns:
    rad = np.deg2rad(df['WindDirection'])
    sin = pd.Series(np.sin(rad), index=df.index)
    cos = pd.Series(np.cos(rad), index=df.index)
    sin_i = sin.interpolate(method='time', limit_direction='both')
    cos_i = cos.interpolate(method='time', limit_direction='both')
    ang = np.arctan2(sin_i, cos_i)
    df['WindDirection'] = (np.rad2deg(ang) % 360.0)

# interpolate other non-circular continous colums
for col in [c for c in cont_cols if c != 'WindDirection']:
    df[col] = df[col].astype(float).interpolate(method='time', limit_direction='both')
# To return standard integer after intepolation, reset index back to default
df = df.reset_index(drop=True)

# Fill any missing gap with median incase there was where interpolation could not reach
for col in cont_cols:
    if col in df.columns:
        med = df[col].median(skipna=True)
        df[col] = df[col].fillna(med)
```

Figure 4.7: Circular Interpolation of WindDirection column.

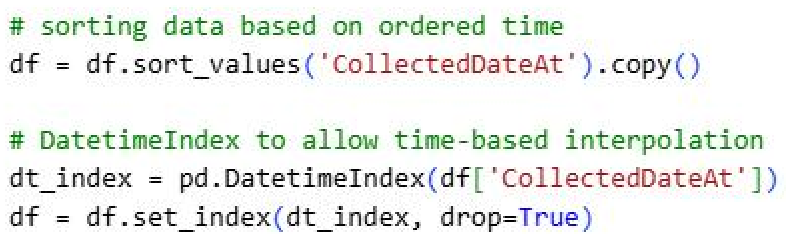
Z-score normalization was performed on selected numerical variables to support outlier detection. This created new columns for the selected variables as shown in the figure 4.8 below. Z-score normalization transforms each value based on the mean and standard deviation of its distribution, enabling a standardized measure of how far a data point deviates from the average.



```python
# create new columns of the variables below which are z-score normalized for outlier detection
for col in [c for c in ['WindSpeed','AirTemperature','Humidity','BarometricPressure','SRAD'] if c in df.columns]:
    m = df[col].mean(); s = df[col].std(ddof=0)
    if s and s > 0:
        df[col + '_z'] = (df[col] - m) / s
```

Figure 4.8: Z-score Normalization of Numerical Variables.

Finally, CollectedDateAt column was sorted based on ordered time to detect missing values for given timestamps and converted to pandas DatetimeIndex to allow time-based interpolation. The process was handled with few lines of code as shown in figure 4.9 below.



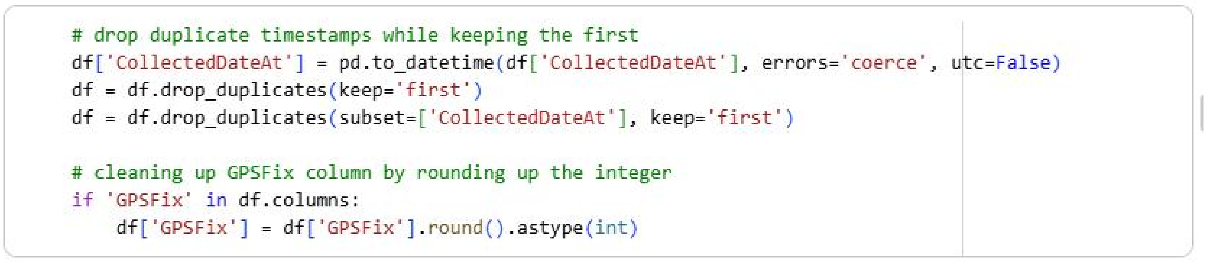
```python
# sorting data based on ordered time
df = df.sort_values('CollectedDateAt').copy()

# DatetimeIndex to allow time-based interpolation
dt_index = pd.DatetimeIndex(df['CollectedDateAt'])
df = df.set_index(dt_index, drop=True)
```

Figure 4.9: Time-based Interpolation.

4.2.4 Data Consistency

Consistency was evaluated with respect to time and type. The date time field, CollectedDateAt, was changed to a standard datetime configuration, and redundant timestamps were purged to eliminate timestamps value duplication.

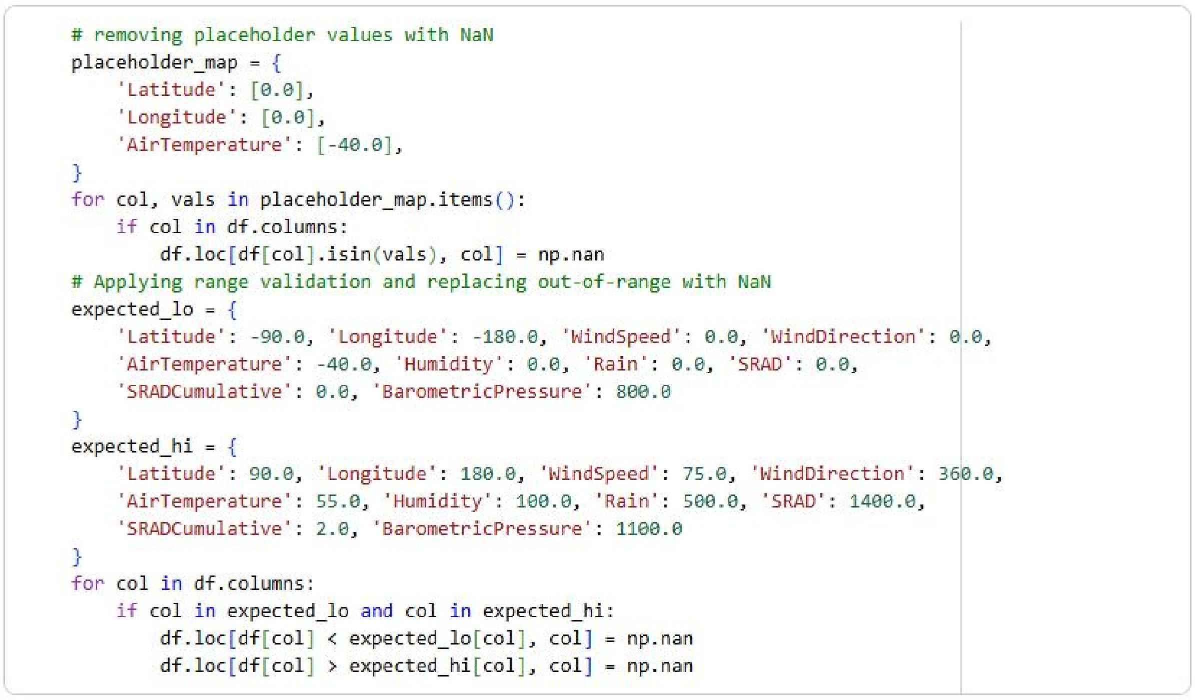
Two duplicate removal operations occurred in the data cleaning workflow. The first command, df = df.drop_duplicates(keep='first'), deletes completely identical rows or complete duplicates which could result from errors in export, a copy of the sensor logs, or any duplication error in the file handling. Removing these will only keep one copy for the duplicate observation. The second command, df = df.drop_duplicates(subset=['CollectedDateAt'], keep='first') handles the duplicates of timestamps. Automated weather stations in time-series meteorological datasets often generate duplicate readings at the same time stamp. This is a common occurrence that may happen due to, for example, buffering, a reset of a sensor, or a communication delay. These duplications if retained would prejudice the temporal interpolation, indexing and modelling assumptions. As such, the two commands would remove both exact row duplicates and duplicate at the timestamp level while keeping the first data. Thus, the dataset would be internally consistent and suitably structured for time-series analysis and machine-learning modelling.



```python
# drop duplicate timestamps while keeping the first
df['CollectedDateAt'] = pd.to_datetime(df['CollectedDateAt'], errors='coerce', utc=False)
df = df.drop_duplicates(keep='first')
df = df.drop_duplicates(subset=['CollectedDateAt'], keep='first')

# cleaning up GPSFix column by rounding up the integer
if 'GPSFix' in df.columns:
    df['GPSFix'] = df['GPSFix'].round().astype(int)
```

Figure 4.10: Row and Timestamps Duplication Handling.

For coherence, GPSFix column was rounded and converted to an integer type as seen in figure 4.10 above. With these actions, all temporal metrics were properly aligned and categorical variables were brought to the same standard, ensuring coherent interpretation of the entire dataset.
Finally, known placeholder values for Latitutude, Longitude and AirTemperature columns (such as 0.0, -40) were replaced with missing values (NAN) before analysis to prevent them from being interpreted as genuine measurements. Removing placeholders prevents the model from learning incorrect patterns or biases caused by invalid values artificially inserted by sensors or data-entry systems.



```python
# removing placeholder values with NaN
placeholder_map = {
    'Latitude': [0.0],
    'Longitude': [0.0],
    'AirTemperature': [-40.0],
}
for col, vals in placeholder_map.items():
    if col in df.columns:
        df.loc[df[col].isin(vals), col] = np.nan
# Applying range validation and replacing out-of-range with NaN
expected_lo = {
    'Latitude': -90.0, 'Longitude': -180.0, 'WindSpeed': 0.0, 'WindDirection': 0.0,
    'AirTemperature': -40.0, 'Humidity': 0.0, 'Rain': 0.0, 'SRAD': 0.0,
    'SRADCumulative': 0.0, 'BarometricPressure': 800.0
}
expected_hi = {
    'Latitude': 90.0, 'Longitude': 180.0, 'WindSpeed': 75.0, 'WindDirection': 360.0,
    'AirTemperature': 55.0, 'Humidity': 100.0, 'Rain': 500.0, 'SRAD': 1400.0,
    'SRADCumulative': 2.0, 'BarometricPressure': 1100.0
}
for col in df.columns:
    if col in expected_lo and col in expected_hi:
        df.loc[df[col] < expected_lo[col], col] = np.nan
        df.loc[df[col] > expected_hi[col], col] = np.nan
```

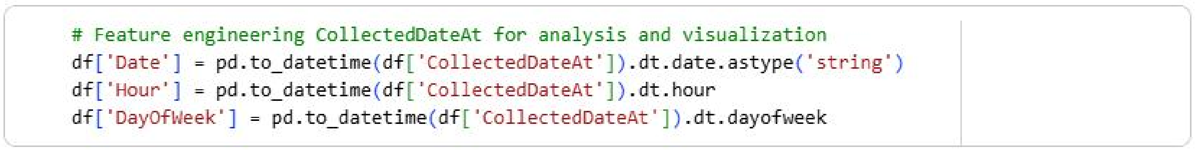
Figure 4.11: Placeholder Values Correction and Out-of-Range Values Definition.

As seen from figure 4.11 above, Out-of-Range values were filtered out by defining physical ranges of values expected from the high and low variations of Latitude, Longitude, WindSpeed, WindDirection, AirTemperature, Humidity, Rain, SRAD, SRADCumulative, and BarometricPressure columns. Range validation ensures that every value in the defined columns falls within a scientifically reasonable interval. Values outside these bounds are considered erroneous, often caused by sensor faults, data transmission errors, or environmental noise. Two dictionaries define the minimum (expected_lo) and maximum (expected_hi) acceptable values for each variable and values below the minimum or above the maximum is replaced with NaN.

This step enforces logical consistency, ensuring that the dataset does not contain values that violate domain knowledge which can severely distort model training, leading to inaccurate predictions and poor generalization.

4.2.5 Data Diversity

With respect to data diversity, the temporal features of the dataset were expanded to provide more information, enhancing clarity and generalization. Specifically, temporal variables Date, Hour, and DayOfWeek were extracted from the CollectedDateAt column. The new variables allow more detailed analyses of temporal structures, thereby enhancing the dataset to offer the possibility of studying diurnal and weekly cycles, or environmental trends interpreted over time.



```
# Feature engineering CollectedDateAt for analysis and visualization
df['Date'] = pd.to_datetime(df['CollectedDateAt']).dt.date.astype('string')
df['Hour'] = pd.to_datetime(df['CollectedDateAt']).dt.hour
df['DayOfWeek'] = pd.to_datetime(df['CollectedDateAt']).dt.dayofweek
```

Figure 4.12: Data Diversity for the CollectedDateAt Column.

As depicted in figure 4.12 above, three new features that represent time were created from the CollectedDateAt column through feature engineering. These features can help study delay dependent behavior.

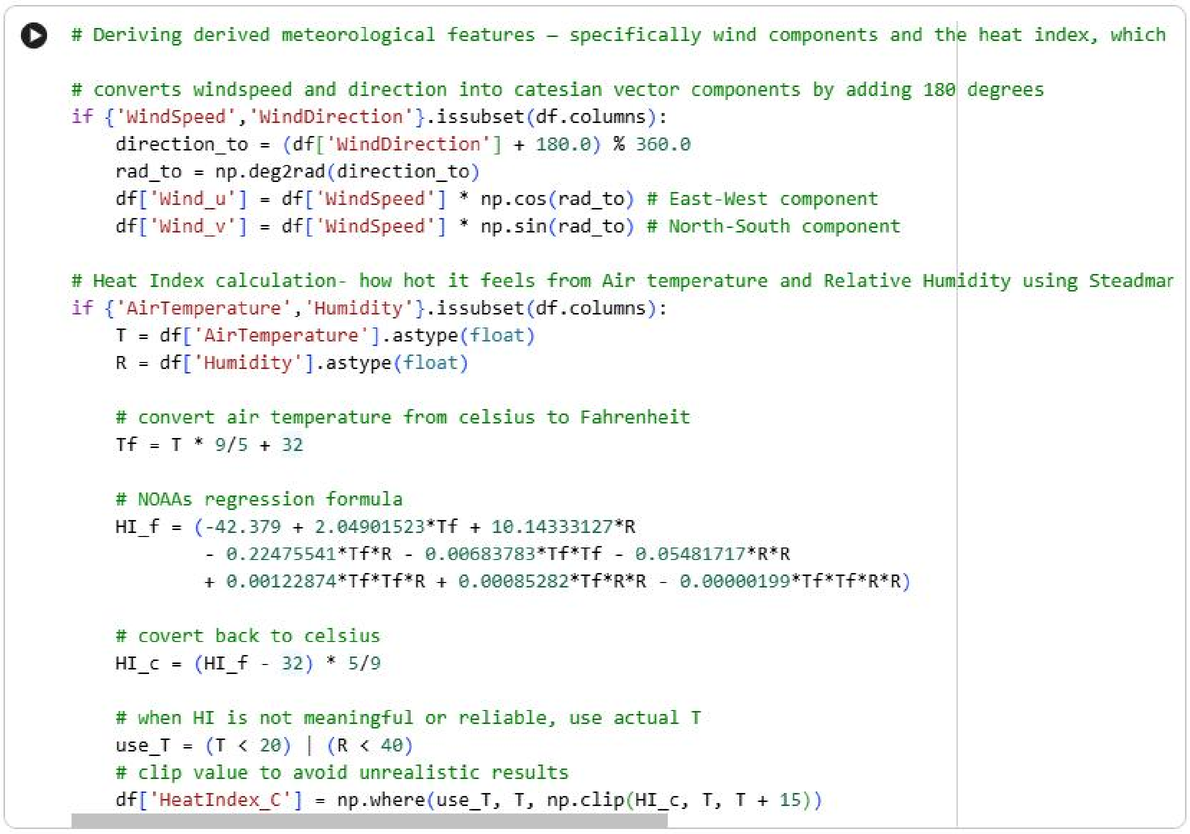
- First, a Date variable was obtained which has the calendar day independent of the clock time so that the daily trends can be analyzed and aggregated.
- Secondly, the Hour variable was created to capture intra-day variation. This is relevant for modelling diurnal cycles in temperature, radiation, and atmospheric pressure.
- Lastly, a DayOfWeek indicator was created to reflect the pattern and seasonality present on a weekly periodicity.

The variables were created using the datetime function of Pandas (df['Date']) which stores the date as a string, df['Hour'] stores the hour of measurement as an integer (0–23), and df['DayOfWeek'] stores the index of the weekday (0 = Monday, 6 = Sunday). All these expanded time features add variety to the dataset and help encourage downstream models to pick up patterns in the target variable (barometric pressure) that are cyclical, calendar-based, and time-of-day specific.

4.2.6 Robustness to Noise

This section of the preprocessing implementation pipeline generates additional meteorological features, specifically wind vector components and the heat index, which provide richer environmental information for subsequent analysis and machine learning modeling. These derived variables are widely used in atmospheric science, environmental monitoring, and human comfort assessment, making them valuable predictors in data-driven systems.

- The wind speed and direction data were expanded to include the vector components of the wind as a more directional dynamic approach. To compute the vector representing the direction toward movement, 180° is added and the angle is converted from degrees to radians using trigonometric decomposition (u= Vcos(θ) and v=Vsin(θ)).
- The standard meteorological formula combining air temperature and humidity creates a Heat Index (HeatIndex_C) and provides a more accurate measure of thermal comfort than temperature alone using National Oceanic Atmospheric Administration (NOAA's) regression.



```
# Deriving derived meteorological features – specifically wind components and the heat index, which

# converts windspeed and direction into catesian vector components by adding 180 degrees
if {'WindSpeed','WindDirection'}.issubset(df.columns):
    direction_to = (df['WindDirection'] + 180.0) % 360.0
    rad_to = np.deg2rad(direction_to)
    df['Wind_u'] = df['WindSpeed'] * np.cos(rad_to) # East-West component
    df['Wind_v'] = df['WindSpeed'] * np.sin(rad_to) # North-South component

# Heat Index calculation- how hot it feels from Air temperature and Relative Humidity using Steadmar
if {'AirTemperature','Humidity'}.issubset(df.columns):
    T = df['AirTemperature'].astype(float)
    R = df['Humidity'].astype(float)

    # convert air temperature from celsius to Fahrenheit
    Tf = T * 9/5 + 32

    # NOAAs regression formula
    HI_f = (-42.379 + 2.04901523*Tf + 10.14333127*R
            - 0.22475541*Tf*R - 0.00683783*Tf*Tf - 0.05481717*R*R
            + 0.00122874*Tf*Tf*R + 0.00085282*Tf*R*R - 0.00000199*Tf*Tf*R*R)

    # covert back to celsius
    HI_c = (HI_f - 32) * 5/9

    # when HI is not meaningful or reliable, use actual T
    use_T = (T < 20) | (R < 40)
    # clip value to avoid unrealistic results
    df['HeatIndex_C'] = np.where(use_T, T, np.clip(HI_c, T, T + 15))
```

Figure 4.13: Data Robustness to Noise Validation.

Two robustness-based transformations on the dataset were carried out for improved applicability for meteorological modelling of wind vector decomposition and Heat Index (HI) computation.

- First, as seen in figure 4.13, the scalar wind reading was transformed into orthogonal u−v components to prevent the discontinuity caused by the circular wrapping of wind direction (0° ≈ 360°). This transformation produces smoother and more directional consistent predictors often employed in atmospheric science. To convert wind direction from meteorological convention to mathematical convention, the wind direction was rotated by 180°, converted the degrees to radian and applied cosine and sine to generate u and v components.
- Also, a Heat Index (HI) feature was created to model the perceived temperature as a function of AirTemperature and Humidity through the Rothfusz regression equation as the basis of NOAA’s heat index formula. From figure 4.13, the heat index is only meaningful when Temperature ≥ 20°C and Humidity ≥ 40%. Outside these conditions, the actual air temperature is used instead.

These transformations enhance the dataset with more relevant and model-friendly features which enhances robustness and interpretability.

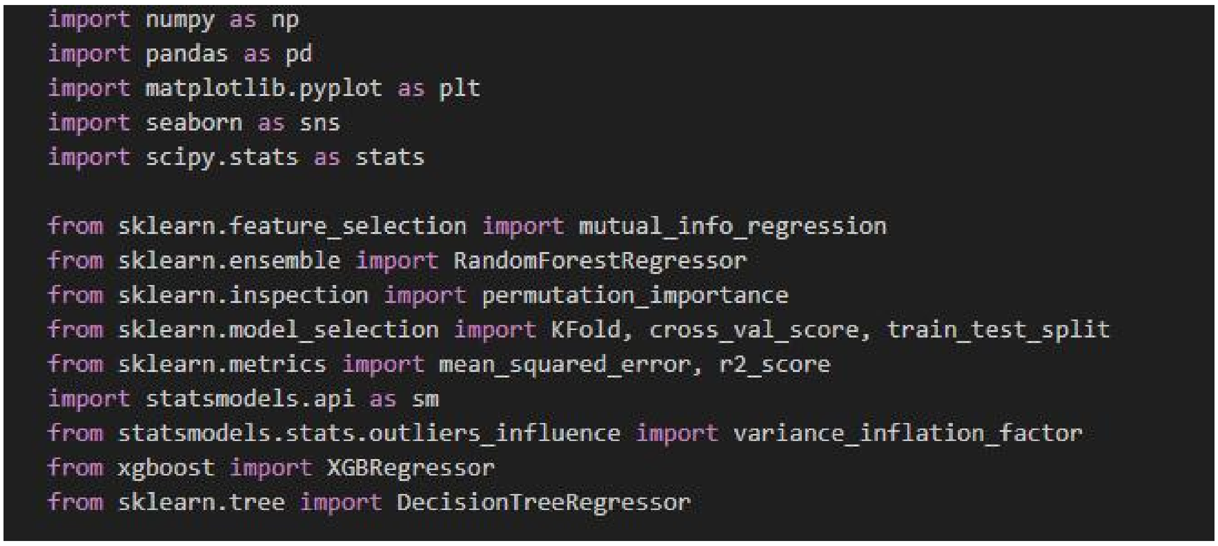
After cleaning, correcting, interpolating, and engineering features, the obtained improved dataset was exported to an Excel file (Cleaned_file). It has a total of 28,446 rows and 23 variables, which will ensure a stable, reproducible version for subsequent Exploratory Data Analysis and modelling. The export was performed using the following code:

```
with pd.ExcelWriter("Clean_file.xlsx", engine="xlsxwriter") as writer:
    df.to_excel(writer, index=False, sheet_name='clean')
```

4.3 Exploratory Data Analysis (EDA) Implementation

In order to prepare the dataset for the subsequent stages of completeness analysis and improvement, we concentrated on finding trends and insights in the dataset (Clean_file.xlsx) during the Exploratory Data Analysis (EDA) phase. The EDA procedure involves importing the improved (Clean_file.xlsx) dataset, generating

important visualizations, and evaluating the findings to discover possible issues and patterns.
The dataset was imported using Python's Pandas package, allowing efficient handling and preparation of the data. The EDA was performed by importing popular python data science packages like NumPy, Pandas, Matplotlib, Seaborn, SciPy, scikit-learn. These libraries help with statistical computing, visualization, feature testing, model assessment and multicollinearity. XGBoost was also imported for feature importance comparisons, while StatsModels was used to calculate the Variance Inflation Factor (VIFs). For the purpose of EDA, target variable was set to be BarometricPressure on which further univariate analysis, correlation, outlier detection and multicollinearity analysis was performed.



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats

from sklearn.feature_selection import mutual_info_regression
from sklearn.ensemble import RandomForestRegressor
from sklearn.inspection import permutation_importance
from sklearn.model_selection import KFold, cross_val_score, train_test_split
from sklearn.metrics import mean_squared_error, r2_score
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from xgboost import XGBRegressor
from sklearn.tree import DecisionTreeRegressor
```

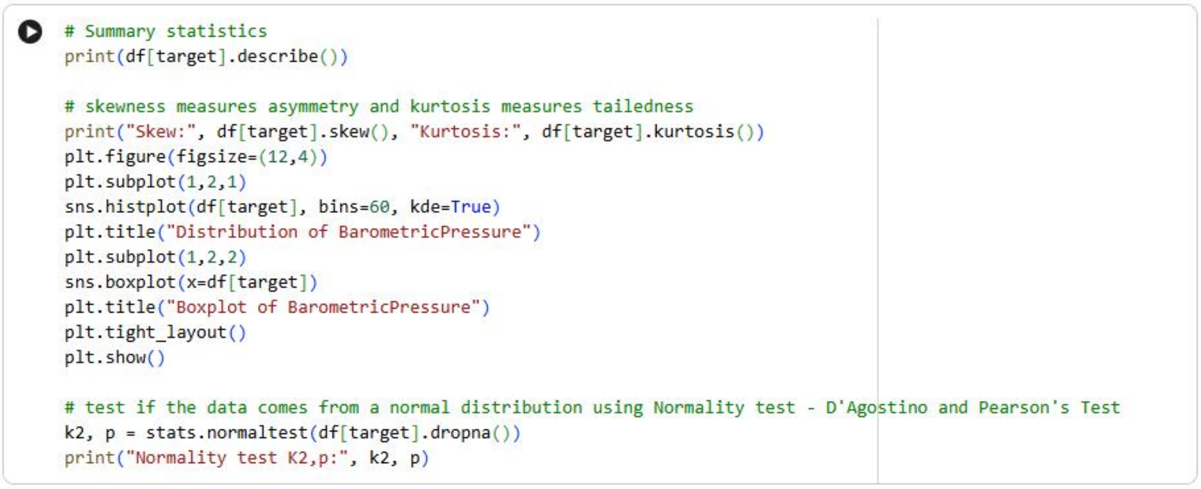
Figure 4.14: Exploratory Data Analysis Libraries.

4.3.1 Target Variable Distribution Analysis

This section discusses the different Exploratory Data Analysis transformations and checks that was performed on the improved Clean_file.xlsx dataset to prepare for modeling while focusing on the target variable (BarometricPressure).

Univariate Analysis using Skewness and Kurtosis

Univariate analysis was performed on the target variable (BarometricPressure) to visualize the distribution analysis by calculating the skewness and kurtosis. Skewness measures the asymmetry of target variable in the distribution. A positive skew value means long tail on the right while a negative skew means long tail on the left. On the other hand, Kurtosis measures the heaviness of the tail relative to a normal distribution. Kurtosis values >= 3 represents heavier tails and outliers (leptokurtic) while lower values < 3 represents lighter tails.



```
# Summary statistics
print(df[target].describe())

# skewness measures asymmetry and kurtosis measures tailedness
print("Skew:", df[target].skew(), "Kurtosis:", df[target].kurtosis())
plt.figure(figsize=(12,4))
plt.subplot(1,2,1)
sns.histplot(df[target], bins=60, kde=True)
plt.title("Distribution of BarometricPressure")
plt.subplot(1,2,2)
sns.boxplot(x=df[target])
plt.title("Boxplot of BarometricPressure")
plt.tight_layout()
plt.show()

# test if the data comes from a normal distribution using Normality test - D'Agostino and Pearson's Test
k2, p = stats.normaltest(df[target].dropna())
print("Normality test K2,p:", k2, p)
```

Figure 4.15: Skewness and Kurtosis Deviation Calculation.

As depicted by figure 4.15 above, the skewness and kurtosis deviation analysis of the target variable was performed to find out how much the target variable deviated from normality. To test the normality formally, the D’Agostino–Pearson omnibus test was used. The test statistic which measures departure from normality based on skewness and kurtosis, K2 ≈ 8698.78 yielded p = 0.0, strongly rejecting normality.

From this test, showing that the target variable rejected normality, the use of robust algorithms for non-normality (tree-based models) decision was drawn. The result of the analysis was visualized using histograms from pandas mathplotlib and boxplot from seaborn libraries as shown below.

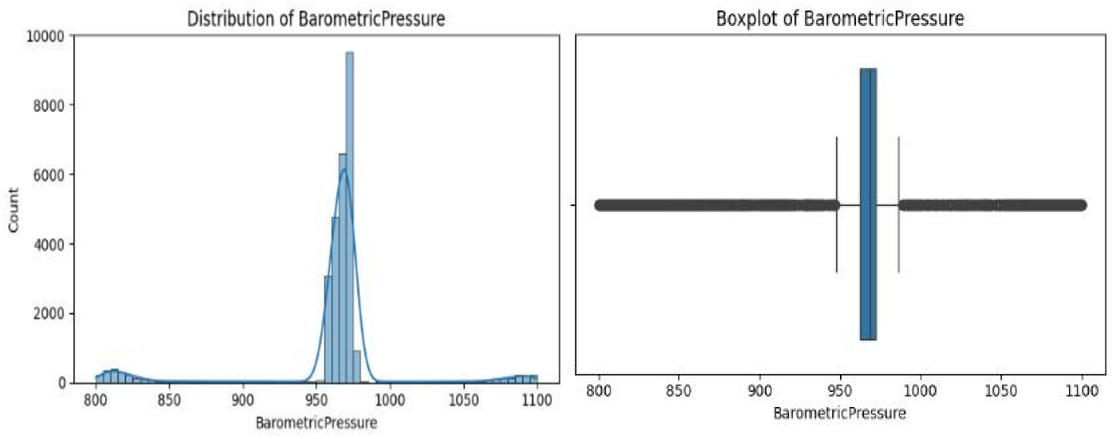


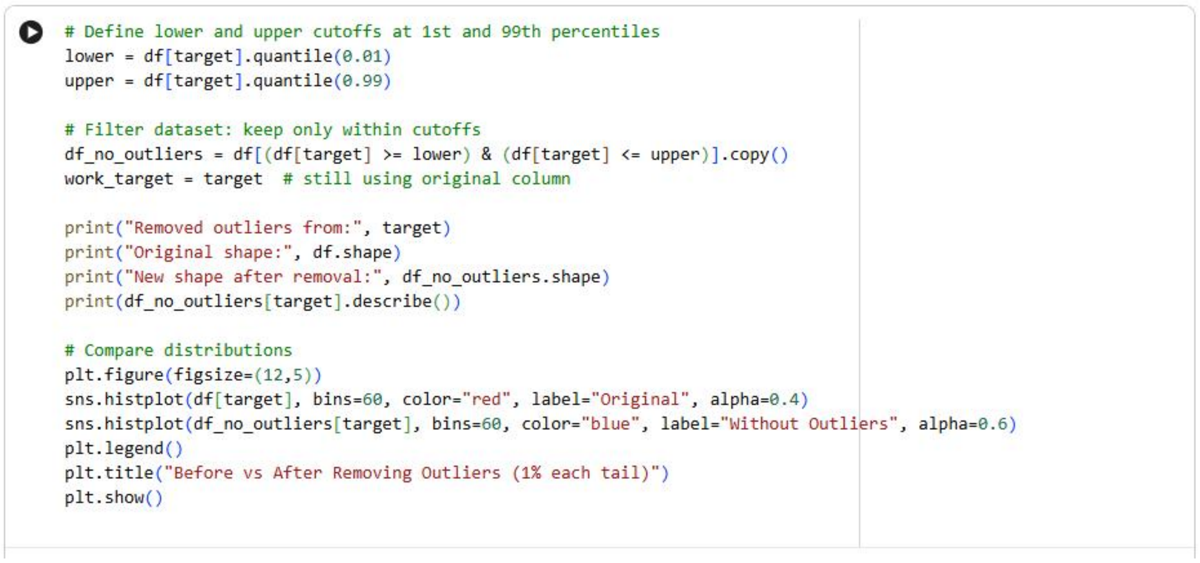
Figure 4.16: Skewness and Kurtosis Analysis Visualization.

These metrics helped evaluate whether the target data deviated from normality, which may affect model performance and guided preprocessing decisions such as transformations or outlier handling. The results of these analysis will be discussed in the next chapter.

4.3.2 Outlier Detection Implementation

Following the results of univariate analysis, outlier detection and winsorisation was performed to handle outliers that may have been detected in the distribution analysis. According to univariate analysis, distribution was heavily tailed with many extremes in both lower (≈800 hPa) and upper (≈1100 hPa) sides. Specifically, a trimming approach based on percentiles was applied to the target variable to cut-off the lower and upper thresholds known as the 1st and 99th percentiles respectively [61].

This created a data frame without outliers (df_no_outliers) by removing ranges falling out of the threshold but retaining the overall form of the central distribution.



```
# Define lower and upper cutoffs at 1st and 99th percentiles
lower = df[target].quantile(0.01)
upper = df[target].quantile(0.99)

# Filter dataset: keep only within cutoffs
df_no_outliers = df[(df[target] >= lower) & (df[target] <= upper)].copy()
work_target = target  # still using original column

print("Removed outliers from:", target)
print("Original shape:", df.shape)
print("New shape after removal:", df_no_outliers.shape)
print(df_no_outliers[target].describe())

# Compare distributions
plt.figure(figsize=(12,5))
sns.histplot(df[target], bins=60, color="red", label="Original", alpha=0.4)
sns.histplot(df_no_outliers[target], bins=60, color="blue", label="Without Outliers", alpha=0.6)
plt.legend()
plt.title("Before vs After Removing Outliers (1% each tail)")
plt.show()
```

Figure 4.17: Outlier Detection and Winsorisation.

After eliminating the extreme 1% and 99% tails, the dataset size reduced to 27876 from 28446, which means 570 extreme readings were removed. The new distribution is more compact and symmetric in the center, with high pressure and low pressure outliers having less influence. The target variable therefore becomes more stable and reliable for modelling which can be seen from the difference in the normal distribution and the distribution without outliers as plotted below.

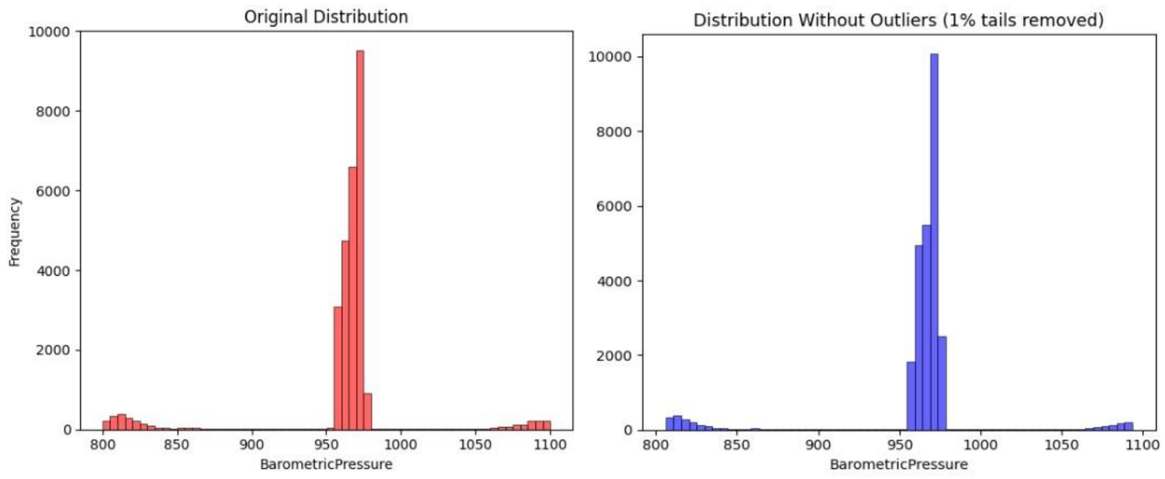


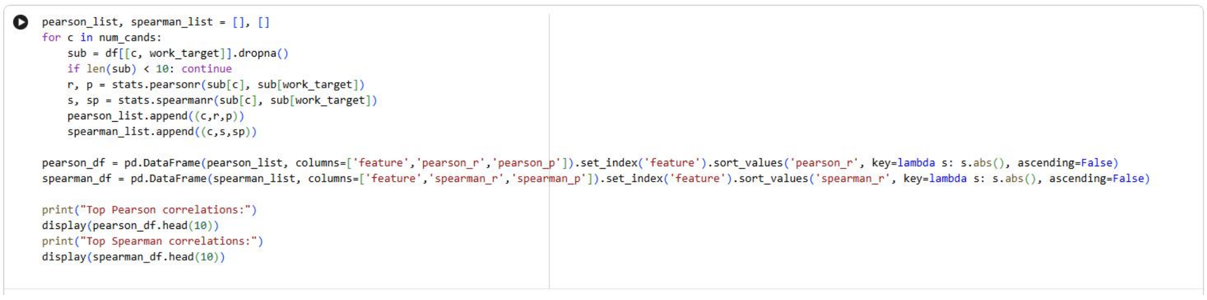
Figure 4.18: Before and After Outlier Removal Distribution.

4.3.3 Predictors and Correlation Implementation

This section discusses the implementation of correlation analysis of both linear and non linear relationships between numerical features (num_cands) and the target variable. In this research, the Pearson and Spearman correlations was employed to implement the linear and monotonic associations respectively while mutual information implemented the correlation between non-linear relationships.

Linear and Monotonic Relationship Analysis

This section implements and ranks Pearson and Spearman correlation coefficients between a list of numerical candidate features (num_cands) and the target variable BarometricPressure (work_target). The goal is to understand which features are most strongly related to the target variable, either linearly or monotonically, a critical step in feature selection and model design.



```
pearson_list, spearman_list = [], []
for c in num_cands:
    sub = df[[c, work_target]].dropna()
    if len(sub) < 10: continue
    r, p = stats.pearsonr(sub[c], sub[work_target])
    s, sp = stats.spearmanr(sub[c], sub[work_target])
    pearson_list.append((c,r,p))
    spearman_list.append((c,s,sp))

pearson_df = pd.DataFrame(pearson_list, columns=['feature','pearson_r','pearson_p']).set_index('feature').sort_values('pearson_r', key=lambda s: s.abs(), ascending=False)
spearman_df = pd.DataFrame(spearman_list, columns=['feature','spearman_r','spearman_p']).set_index('feature').sort_values('spearman_r', key=lambda s: s.abs(), ascending=False)

print("Top Pearson correlations:")
display(pearson_df.head(10))
print("Top Spearman correlations:")
display(spearman_df.head(10))
```

Figure 4.19: Linear Correlation Implementation on Target Variable.

Before making the calculations, non-predictor fields like the timestamp, capped pressure values, and the categorical representations were stripped off the candidate.

Only continuous numerical features were selected to ensure statistical validity. The code calculates both Pearson and Spearman correlations between selected numerical features and the target variable, BarometricPressure. Pearson measures linear relationships, while Spearman captures monotonic (including nonlinear) patterns. This helps identify the most influential variables, guiding effective feature selection and improving the overall modeling process. Missing values for each predictor was excluded using the `dropna()` function during the calculation of correlation coefficient (r) and p-value to ensure completeness. After computing each of Pearson and Spearman correlation coefficients (r) and their significance (p), the features are ranked in descending order, based on the absolute strength of their correlation with the target variable as seen in figure 4.19 above.

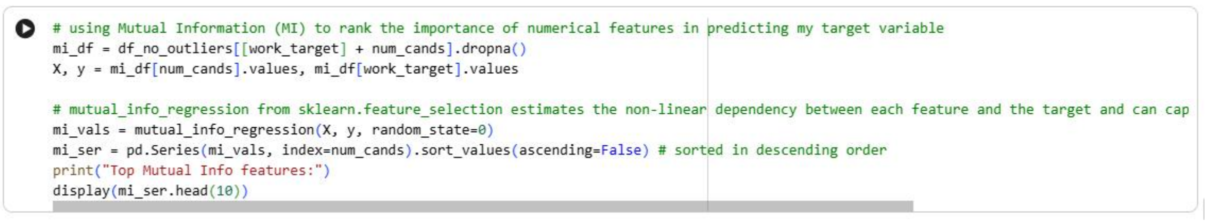
The Pearson correlation results showed that Humidity feature has the strongest positive linear relationship with BarometricPressure with a value of 0.2418 while AirTemperature and HeatIndex_C exhibited inverse linear relationships with BarometricPressure given the value -0.1711 & -0.1249 respectively.

Spearman correlation result showed that HeatIndex_C and AirTemperature have a strong negative monothonic relationship with BarometricPressure with values -0.6017 & -0.5981 respectively while Humidity showed the strongest positive monothonic relationship with BarometricPresure with value 0.1652. The results of these correlation coefficients (r) and their significance (p) will be discussed in the next chapter.

Non-Linear Relationship Analysis

Although Pearson and Spearman correlations detect linear and monotonic associations respectively, many processes exhibit non-linear and multi-dimensional dependency structures, which correlation in its direct form cannot capture. For the implementation of non-linear relationship correlation, Mutual Information (MI) was employed to capture non-linear dependencies.

Mutual Information (MI) was achieved using scikit-learn’s mutual_info_regression library on all numerical candidate predictors (num_cands) and the target variable (work_target) with the implementation code shown below:



```
# using Mutual Information (MI) to rank the importance of numerical features in predicting my target variable
mi_df = df_no_outliers[[work_target] + num_cands].dropna()
X, y = mi_df[num_cands].values, mi_df[work_target].values

# mutual_info_regression from sklearn.feature_selection estimates the non-linear dependency between each feature and the target and can cap
mi_vals = mutual_info_regression(X, y, random_state=0)
mi_ser = pd.Series(mi_vals, index=num_cands).sort_values(ascending=False) # sorted in descending order
print("Top Mutual Info features:")
display(mi_ser.head(10))
```

Figure 4.20: Non-Linear Correlation Implementation on Target Variable.

The Mutual Information (MI) results reveal several important findings of how each numerical feature (num_cands) associate non-linearly to BarometricPressure. It showed that temperature-driven variables dominate non-linear influence as AirTemperature and HeatIndex_C showed strong MI scores indicating a strong positive non-linear relationship with BarometricPressure with value 0.864. Radiation effects exhibited non-linear contribution as the MI values of SRADCumulative and SRAD are significant.

The Mutual Information values reveal more complex dependencies than Pearson or Spearman correlations. It shows how a number of predictors affect BarometricPressure in non-linear ways. These findings support the use of tree-based models (Random Forest, Decision Tree and XGBoost) in the subsequent stage of modelling because they can easily detect non-linear interactions that linear models cannot.

Categorical Predictors

The feature engineered temporal features, Hour and DayOfWeekName (created earlier from CollectedDateAt) were treated as categorical predictors to see if BarometricPressure exhibits any systematic patterns over the day and over week. In this section, hourly and daily trends were explored by plotting daily distribution showing BarometricPressure at each hour of the day using boxplots from the seaborn and mathplotlib library.

The resulting hourly plot as seen in figure 4.21 below shows that the median pressure remains relatively stable across hours, with only modest fluctuations. Some hours display slightly wider inter-quartile ranges and more extreme values, suggesting that intra-day variability is present but not dominant.

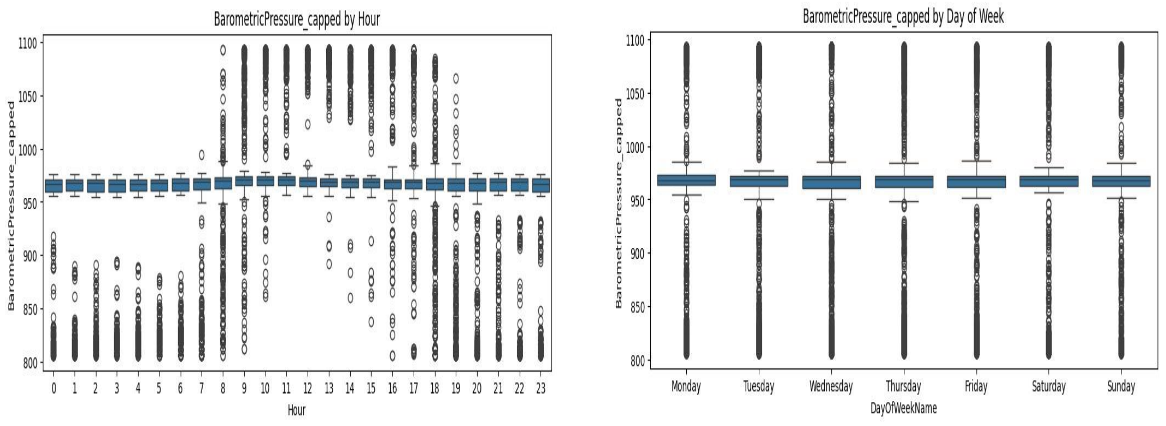
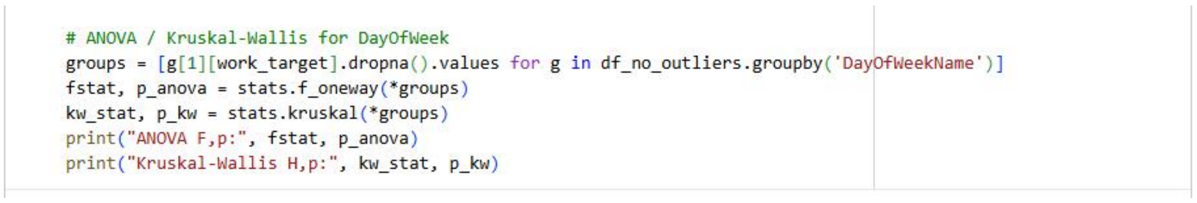


Figure 4.21: Hourly and Daily Distribution of BarometricPressure.

The median is often around the same number for the different days which is also seen from figure 4.21 that the spread for each day is also comparably alike, with slight differences in the observations. To determine if differences between the days exist, a one-way ANOVA and a non-parametric Kruskal-Wallis test was conducted for both mean and distributional differences using the implementation code below:

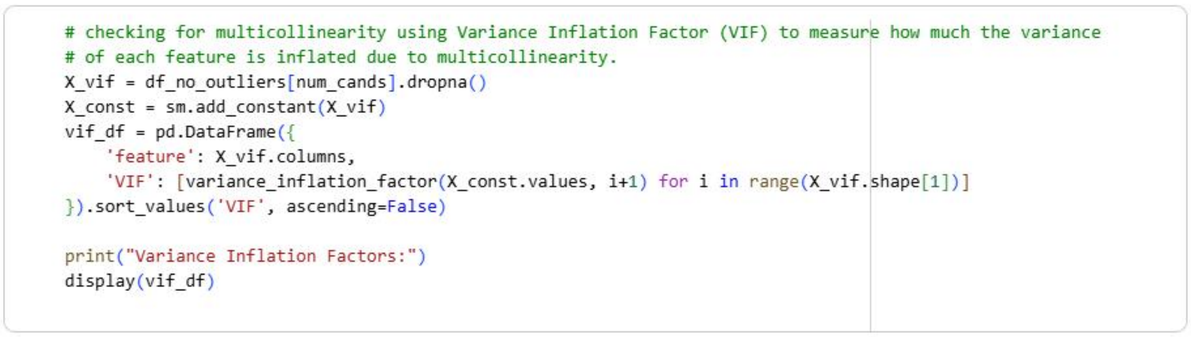


```
# ANOVA / Kruskal-Wallis for DayOfWeek
groups = [g[1][work_target].dropna().values for g in df_no_outliers.groupby('DayOfWeekName')]
fstat, p_anova = stats.f_oneway(*groups)
kw_stat, p_kw = stats.kruskal(*groups)
print("ANOVA F,p:", fstat, p_anova)
print("Kruskal-Wallis H,p:", kw_stat, p_kw)
```

Figure 4.22: ANOVA and Kruskal-Wallis Test.

4.3.4 Variance Inflation Factor (VIF) Implementation

To measure how much the variance of each feature is inflated between the numerical predictors due to multicollinearity, Variance Inflation Factor (VIF) was checked on the `df_no_outliers` data set. VIF measures how much the variance of an estimated regression coefficient increases when predictors are correlated. For this, only numerical candidate variables (num_cands) were included dropping rows with missing values for these predictors using the `dropna()` function.



```
# checking for multicollinearity using Variance Inflation Factor (VIF) to measure how much the variance
# of each feature is inflated due to multicollinearity.
X_vif = df_no_outliers[num_cands].dropna()
X_const = sm.add_constant(X_vif)
vif_df = pd.DataFrame({
    'feature': X_vif.columns,
    'VIF': [variance_inflation_factor(X_const.values, i+1) for i in range(X_vif.shape[1])]
}).sort_values('VIF', ascending=False)

print("Variance Inflation Factors:")
display(vif_df)
```

Figure 4.23: Variance Inflation Factor for Multicollinearity.

The result of multicollinearity using VIF was the basis of the selection of final predictors features that was used to train the models. Using Variance Inflation Factor (VIF) to measure how much the variance of each feature is inflated due to multicollinearity, all features with VIF values of greater than 9 were eliminated. Thus, the following columns were selected as final predictors based on the fact that they have VIF values less than 9: SRADCumulative, Hour, Wind_v, WindDirection, Wind_u, Latitude, Longitude, Rain, DayOfWeek, and GPSFix.

4.4 Baseline Models Implementation

Three supervised regression models were trained on the cleaned and engineered dataset to benchmark the predictive power of tree-based algorithms for the target variable, BarometricPressure. The models include Decision Tree Regressor, Random Forest Regressor, and XGBoost Regressor. The final predictors list was selected as mentioned in section 4.3.4, after the multicollinearity analysis (VIF filtering) to avoid redundant information and unstable coefficients. Datasets were split into training (80%) and validation (20%) sets using train_test_split for each model and feature importance were calculated as well as aggregated.

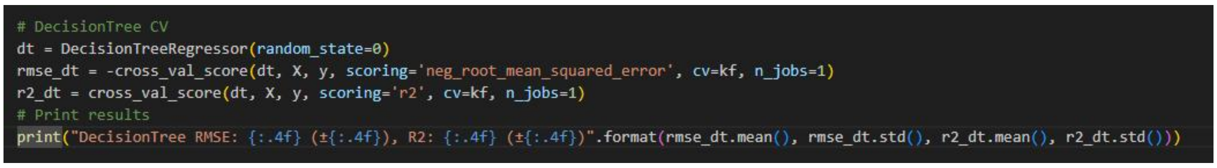
4.4.1 Decision Tree Regressor

The Decision Tree Regressor serves as a baseline model due to its interpretability and ability to capture non-linear relationships. After selecting the final set of predictors based on VIF thresholds, the model is trained using the cleaned dataset with the final predictors where missing values were removed. The implementation uses scikit-learn's DecisionTreeRegressor library with a fixed random seed (random_state=0) to ensure reproducibility. No additional hyperparameter tuning is applied, intentionally preserving its baseline nature.

To evaluate generalization performance, a 5-fold cross-validation scheme is employed with shuffling enabled to avoid temporal or ordering bias in the meteorological data. For each fold, the model is trained on 80% of the data and tested on the remaining 20%. Performance is then assessed using two metrics:

- **Root Mean Squared Error (RMSE):** quantifies average prediction error magnitude.
- **R² Score:** captures the proportion of variance explained by the model.

The cross-validated RMSE and R² values are aggregated as mean and standard deviation, producing an RMSE: 11.3313 (±0.3337), R^2: 0.9241 (±0.0089), to represent stability and robustness as seen in figure 4.24 below.



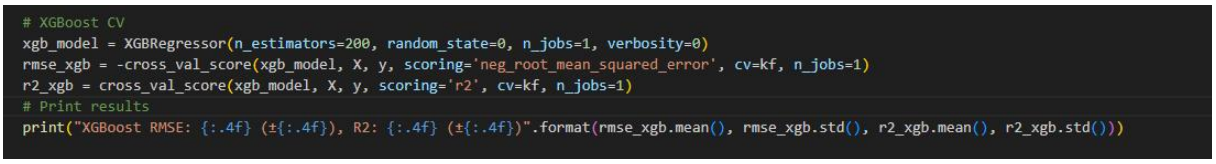
```
# DecisionTree CV
dt = DecisionTreeRegressor(random_state=0)
rmse_dt = -cross_val_score(dt, X, y, scoring='neg_root_mean_squared_error', cv=kf, n_jobs=1)
r2_dt = cross_val_score(dt, X, y, scoring='r2', cv=kf, n_jobs=1)
# Print results
print("DecisionTree RMSE: {:.4f} (±{:.4f}), R2: {:.4f} (±{:.4f})".format(rmse_dt.mean(), rmse_dt.std(), r2_dt.mean(), r2_dt.std()))
```

Figure 4.24: Decision Tree Regressor Implementation.

4.4.2 Extreme Gradient Boost (XGBoost) Algorithm

The implementation uses XGBRegressor from xgboost library with 200 estimators (n_estimators=200), A fixed random seed (random_state=0), ensures reproducibility, and parallelization (n_jobs=-1) accelerates model training. Unlike Random Forest, which builds trees independently, XGBoost builds them sequentially, where each tree focuses on correcting residuals.

A consistent 5-fold cross-validation protocol is applied and the model's performance is evaluated using RMSE and R² metrics with the values, RMSE: 9.1840 (±0.5479), R^2: 0.9504 (±0.0042) using the following implementation code:



```
# XGBoost CV
xgb_model = XGBRegressor(n_estimators=200, random_state=0, n_jobs=1, verbosity=0)
rmse_xgb = -cross_val_score(xgb_model, X, y, scoring='neg_root_mean_squared_error', cv=kf, n_jobs=1)
r2_xgb = cross_val_score(xgb_model, X, y, scoring='r2', cv=kf, n_jobs=1)
# Print results
print("XGBoost RMSE: {:.4f} (±{:.4f}), R2: {:.4f} (±{:.4f})".format(rmse_xgb.mean(), rmse_xgb.std(), r2_xgb.mean(), r2_xgb.std()))
```

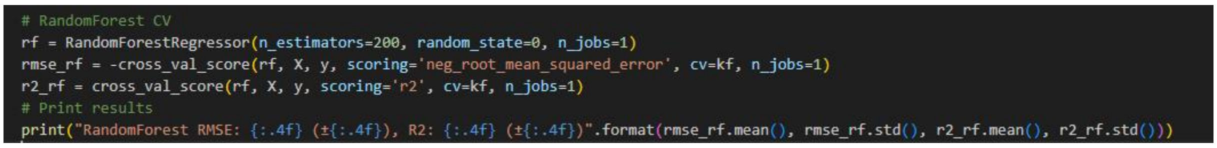
Figure 4.25: XGBoost Implementation.

4.4.3 Random Forest Regressor

The Random Forest Regressor was implemented using RandomForestRegressor from scikit-learn by constructing an ensemble of 200 trees to reduce over-fitting and improve predictive accuracy. Using (n_estimators=200) ensures a stable estimate of feature-target relationships, while (n_jobs=-1) enables parallel computation for efficiency.

The model is evaluated under the same 5-fold cross-validation framework to maintain methodological consistency. In each fold, the forest aggregates predictions from all trees by averaging their outputs, reducing variance compared to a single-tree model. RMSE and R² are again computed for each fold to quantify out-of-sample performance with values, RMSE: 9.0904 (±0.2960), R^2: 0.9514 (±0.0032).

The resulting metrics typically exhibit lower variance across folds due to the ensemble's inherent stability. This implementation demonstrates how Random Forests handle non-linearity.



```
# RandomForest CV
rf = RandomForestRegressor(n_estimators=200, random_state=0, n_jobs=1)
rmse_rf = -cross_val_score(rf, X, y, scoring='neg_root_mean_squared_error', cv=kf, n_jobs=1)
r2_rf = cross_val_score(rf, X, y, scoring='r2', cv=kf, n_jobs=1)
# Print results
print("RandomForest RMSE: {:.4f} (±{:.4f}), R2: {:.4f} (±{:.4f})".format(rmse_rf.mean(), rmse_rf.std(), r2_rf.mean(), r2_rf.std()))
```

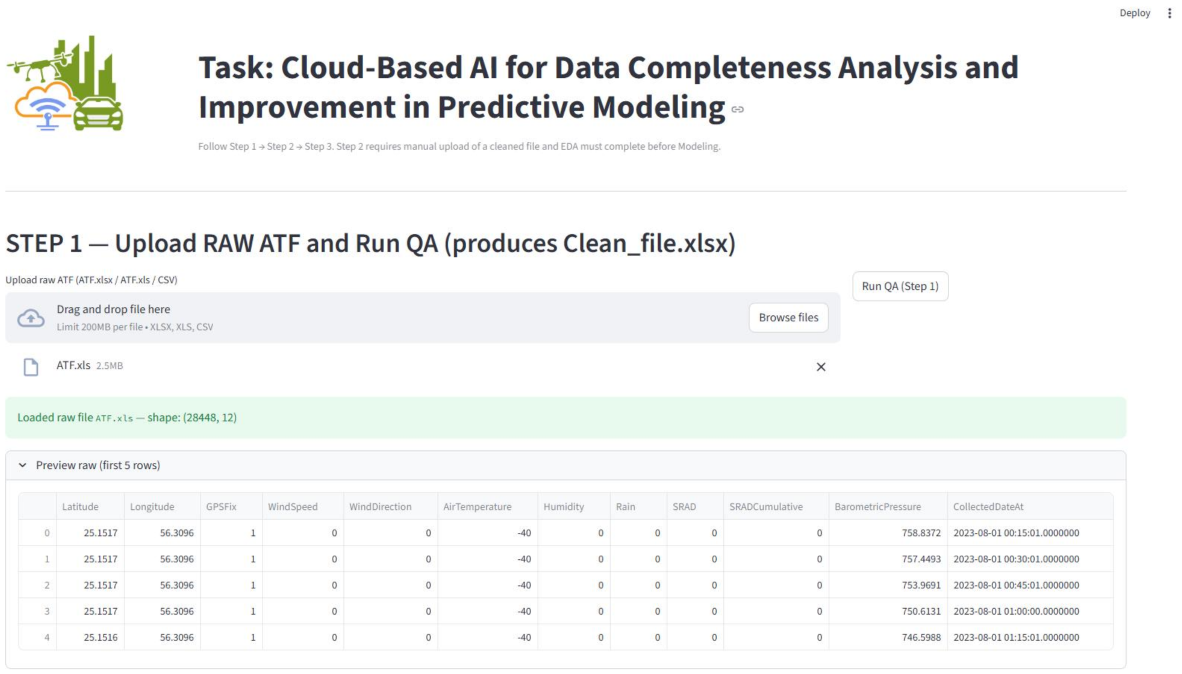
Figure 4.26: Random Forest Regressor Implementation.

4.5 Frontend Integration for Data Completeness Analysis

Following the training and evaluation of the models, the selected prediction models are integrated into an interactive web-based application using the Streamlit framework. Streamlit is a lightweight python framework for quickly converting data science workflows into web applications without the heavy duty front-end coding.The Streamlit integration will provide a web application where the entire data quality verification, exploratory analysis, and predictive modelling can be performed directly from the browser. Built using NumPy, Pandas, SciPy, Statsmodels, and scikit-learn, the Python backend of the system supports all stages of data cleaning, statistical validation, feature engineering and model training acting as an interactive user interface. It follows the work flow of this research with the first handling data initial preprocessing, second step handles Exploratory Data Analysis (EDA) and the last step handle the model training and dashboard results.

Step 1 – Raw File Upload and Automated QA Pipeline

The application begins with a user login page after which loading the original input data for initial preprocessing can be done. The interface of the app is built using `st.file_uploader`. The user can drag and drop the file or upload it manually. When Streamlit detects a file, it reads it back in with Pandas on the spot. The read data is displayed using st.dataframe. This helps to ensure that the user can visually confirm the format, column names and total number of rows. Through st.success, informational banners display that the file has been successfully uploaded and displays its shape. By doing this, it ensures that the rest of the cleaning and transforming code relies on this validated input, paving the way for the rest of the workflow.



Task: Cloud-Based AI for Data Completeness Analysis and Improvement in Predictive Modeling

Follow Step 1 → Step 2 → Step 3. Step 2 requires manual upload of a cleaned file and EDA must complete before Modeling.

STEP 1 — Upload RAW ATF and Run QA (produces Clean_file.xlsx)

Upload raw ATF (ATF.xlsx / ATF.xls / CSV)

Loaded raw file `ATF.xls` — shape: (28448, 12)

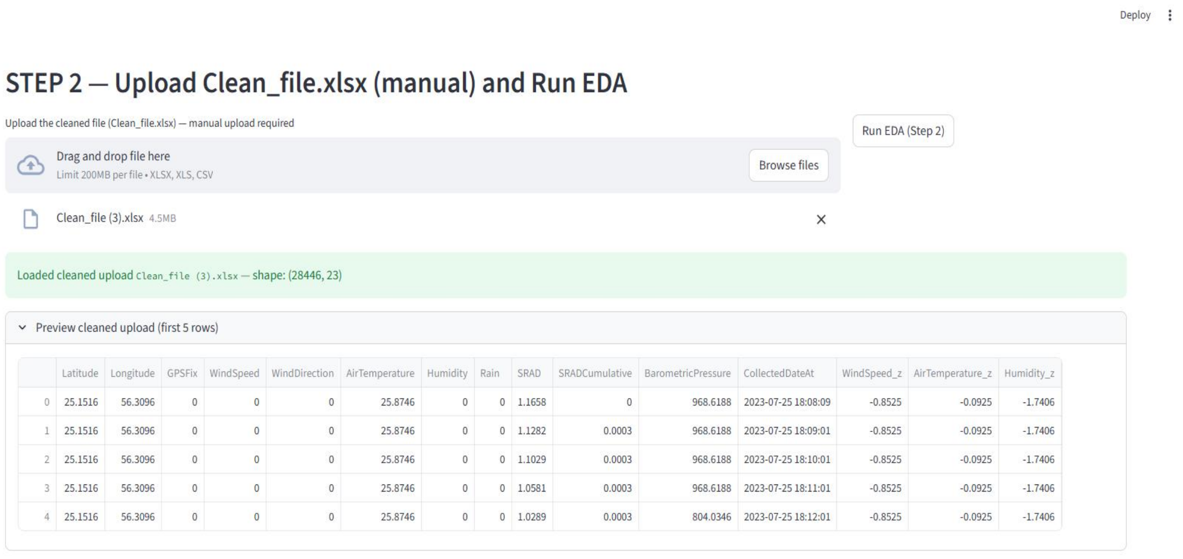
Preview raw (first 5 rows)

	Latitude	Longitude	GPSFix	WindSpeed	WindDirection	AirTemperature	Humidity	Rain	SRAD	SRADCumulative	BarometricPressure	CollectedDateAt
0	25.1517	56.3096	1	0	0	-40	0	0	0	0	758.8372	2023-08-01 00:15:01.0000000
1	25.1517	56.3096	1	0	0	-40	0	0	0	0	757.4493	2023-08-01 00:30:01.0000000
2	25.1517	56.3096	1	0	0	-40	0	0	0	0	753.9691	2023-08-01 00:45:01.0000000
3	25.1517	56.3096	1	0	0	-40	0	0	0	0	750.6131	2023-08-01 01:00:00.0000000
4	25.1516	56.3096	1	0	0	-40	0	0	0	0	746.5988	2023-08-01 01:15:01.0000000

Figure 4.27: Raw File Upload and Automated QA Pipeline.

Step 2 – Uploading Clean_file.xlsx and Running EDA

As soon as the users manually upload the cleaned dataset (Clean_file.xlsx) which was obtained earlier from initial preprocessing, the second step gets activated. Streamlit again relies on st.file_uploader to collect the uploaded file and DataFrame shown to the user using st.dataframe which shows the default first five rows. The “Run EDA (step 2)” button uses st. button and computes all exploratory data analysis implementation steps defined in previous sections of this chapter. When EDA is running, the backend does descriptive statistics, correlation, outlier filtering, feature engineering checks, and predictor screening. The results is shown with Streamlit primitives like st.write, st.table, and st.pyplot to display plots. At the completion of EDA, a message “EDA completed successfully” is displayed. And this completion flag is needed for unlocking the modeling step. Making EDA phase compulsory ensures that model pipeline is not altered thereby enforcing structure.


Deploy
Run EDA (Step 2)
Drag and drop file here
Limit 200MB per file • XLSX, XLS, CSV
Browse files
Clean_file (3).xlsx 4.5MB

STEP 2 — Upload Clean_file.xlsx (manual) and Run EDA

Upload the cleaned file (Clean_file.xlsx) — manual upload required

Loaded cleaned upload `Clean_file (3).xlsx` — shape: (28446, 23)

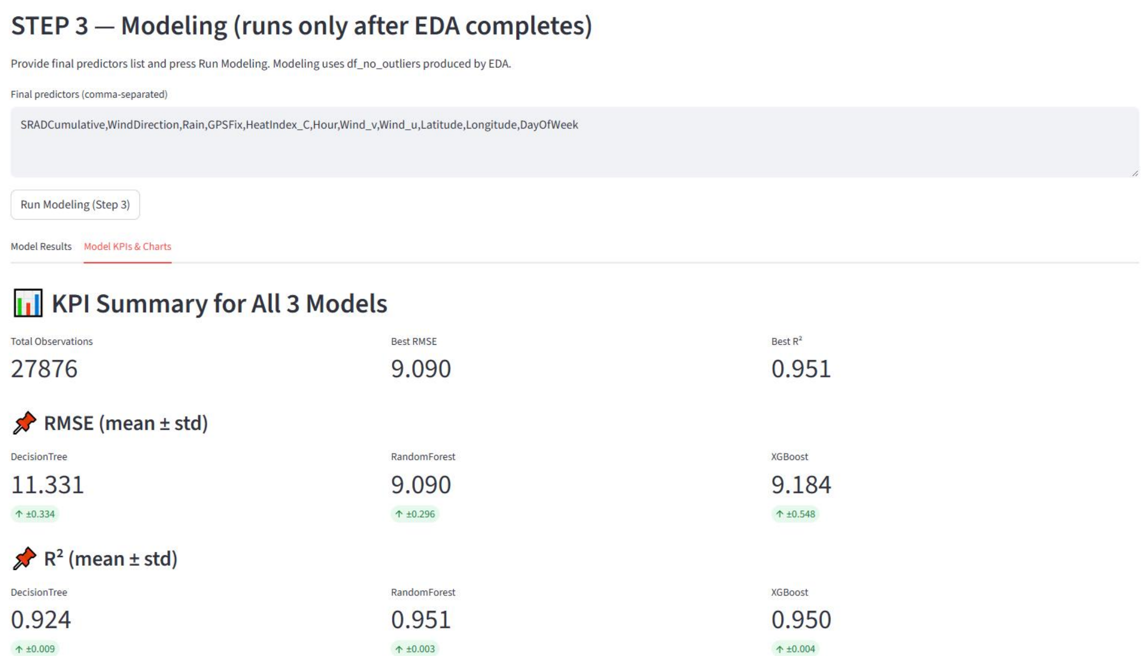
Preview cleaned upload (first 5 rows)

	Latitude	Longitude	GPSFix	WindSpeed	WindDirection	AirTemperature	Humidity	Rain	SRAD	SRADCumulative	BarometricPressure	CollectedDateAt	WindSpeed_z	AirTemperature_z	Humidity_z
0	25.1516	56.3096	0	0	0	25.8746	0	0	1.1658	0	968.6188	2023-07-25 18:08:09	-0.8525	-0.0925	-1.7406
1	25.1516	56.3096	0	0	0	25.8746	0	0	1.1282	0.0003	968.6188	2023-07-25 18:09:01	-0.8525	-0.0925	-1.7406
2	25.1516	56.3096	0	0	0	25.8746	0	0	1.1029	0.0003	968.6188	2023-07-25 18:10:01	-0.8525	-0.0925	-1.7406
3	25.1516	56.3096	0	0	0	25.8746	0	0	1.0581	0.0003	968.6188	2023-07-25 18:11:01	-0.8525	-0.0925	-1.7406
4	25.1516	56.3096	0	0	0	25.8746	0	0	1.0289	0.0003	804.0346	2023-07-25 18:12:01	-0.8525	-0.0925	-1.7406

Figure 4.28: Cleaned File Upload and Running EDA.

Step 3 – Modeling

The last step is to run the predictive modelling algorithms using only the predictors that were generated after the EDA. A text box which is created with st.text_area allows the users to change and confirm the predictor list. “Run Modeling (Step3)” button runs DecisionTree, RandomForest and XGBoost models. With st.write Streamlit shows the RMSE and R² scores for cross-validation. The importances of both model and permutation are shown using st.table for the three models. The combined table that displays the ranking of features renders st.dataframe to allow for wider tables with scroll. The pipeline finishes the process by displaying useful dashboard insights, different plots for visualization & comparison, and provides the ability for downloads.


STEP 3 — Modeling (runs only after EDA completes)

Provide final predictors list and press Run Modeling. Modeling uses df_no_outliers produced by EDA.

Final predictors (comma-separated)

```
SRADCumulative,WindDirection,Rain,GPSFix,HeatIndex_C,Hour,Wind_v,Wind_u,Latitude,Longitude,DayOfWeek
```

KPI Summary for All 3 Models

Total Observations	Best RMSE	Best R^2
27876	9.090	0.951

RMSE (mean ± std)

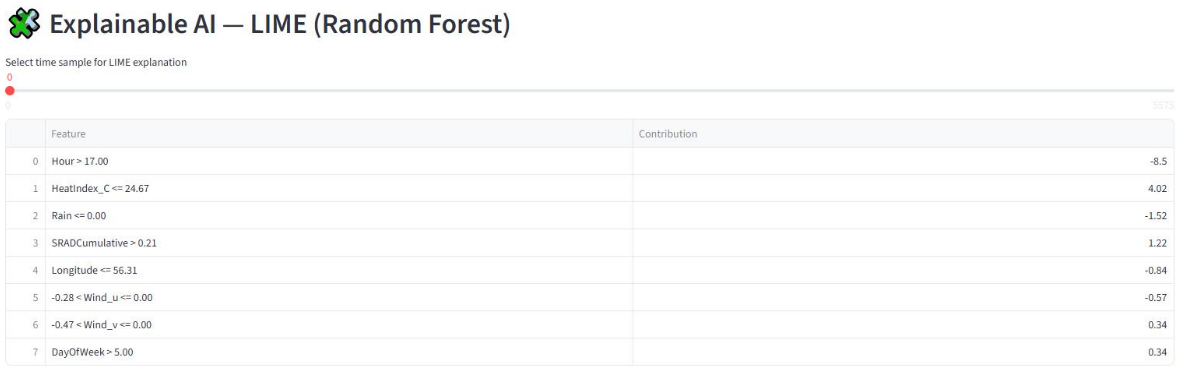
DecisionTree	RandomForest	XGBoost
11.331	9.090	9.184
↑ ±0.334	↑ ±0.296	↑ ±0.548

R^2 (mean ± std)

DecisionTree	RandomForest	XGBoost
0.924	0.951	0.950
↑ ±0.009	↑ ±0.003	↑ ±0.004

Figure 4.29: Running Baseline Models Using Final Predictors.

Lastly, to show how much each variable influenced the prediction using Explainable AI tool, the LIME explanation was performed on Random Forest Regressor model revealing that evening hours (Hour > 17) exert the strongest negative influence on the predicted barometric pressure, while heat index, wind direction, solar radiation, and day of the week provide moderate positive contributions, demonstrating how multiple environmental and temporal factors jointly determine the model's local prediction.


Explainable AI — LIME (Random Forest)

Select time sample for LIME explanation

	Feature	Contribution
0	Hour > 17.00	-8.5
1	HeatIndex_C <= 24.67	4.02
2	Rain <= 0.00	-1.52
3	SRADCumulative > 0.21	1.22
4	Longitude <= 56.31	-0.84
5	-0.28 < Wind_u <= 0.00	-0.57
6	-0.47 < Wind_v <= 0.00	0.34
7	DayOfWeek > 5.00	0.34

Figure 4.30: LIME Local Explanations on Random Forest Regressor Model.

5 Analysis and Results

This chapter presents the results and findings of the framework developed to analyze, improve, and evaluate data completeness effectiveness on model predictions. It outlines the end-to-end workflow, from data ingestion and preprocessing, Exploratory Data Analysis (EDA), completeness checking, dataset improvement, and machine learning evaluation. The goal of this chapter is to present in detail, the results and findings following the implementation steps from the previous chapter.

5.1 Workflow and Process Description

The workflow for the automated steps begins with loading the raw data for initial preprocessing, followed by Exploratory Data Analysis (EDA), and finally, evaluating the improved dataset using machine learning models to measure the effectiveness of the completeness enhancements. All stages of the workflow are automated through Python scripts to ensure repeatability, scalability, and minimal human intervention.

Process Steps in the Workflow

Initial Preprocessing and Completeness Analysis: Systematically evaluates the dataset for incompleteness across multiple dimensions. Automated scripts perform checks for attribute completeness (data types, required fields), time completeness (timestamps, categorical labels), value completeness (missing or null values), value range validity (acceptable bounds for numerical attributes) and overall data quality (duplicates, irregular patterns, anomalous values)
The outcome of this process is an export of an improved cleaned dataset which is fed into EDA in the next step.

Exploratory Data Analysis (EDA): Performs a comprehensive understanding of the quality and completeness of the dataset. This step automatically generates a variety of visualizations, including histograms, scatterplots and boxplots to assess feature distributions. All generated visual outputs are visible on the web dashboard ensuring ease of review. The insights from EDA guide subsequent completeness checks.

Model Evaluation: The Random Forest Regressor, Decision Tree Regressor and XGBoost models are trained on both the raw and cleaned datasets. The workflow automatically computes and compares their key evaluation metrics, Root Mean Squared Error (RMSE) and R^2 value. These metrics provide insight into how much the completeness-enhanced dataset improves predictive performance. The results are displayed in a structured format for interpretation and reporting.

5.2 Data Completeness Results and Analysis

This subsection presents the practical implementation evaluation of the automated data completeness framework developed in this study. It details the results from each stage of the workflow, from data initialization and preprocessing to exploratory data analysis and completeness assessment to dataset improvement and machine learning evaluation. The subsection also reports the results of completeness checks, data quality measurements, and performance comparisons between the raw and enhanced datasets. Through this structured analysis, it is demonstrated that the proposed framework enhances dataset reliability and improves predictive model performance. The automated pipeline in the course of data completeness assessment gained the following results:

1. Skewness and Kurtosis results

Univariate analysis was performed on BarometricPressure to find out how much it deviates from normality using D’Agostino-Pearson normality test based on skewness and kurtosis values.

Summary	Values
Skew	-1.3572
Kurtosis	5.8445
K2	8698.78
p	0.0

Table 5.1: Skew and Kurtosis Results.

Interpretation of Results:

- Negative Skew Value: The distribution is skewed to the left - there is a long tail on the lower pressure side.
- Kurtosis > 3: Leptokurtic heavy tails and outliers, hence more extreme pressures.
- The p-value is zero, which proves the abnormality of the distribution.

In general, barometric pressure was shown in univariate analysis to be highly variable, not normal, and influenced by natural and anomalous extreme levels in the dataset, motivating a later robust modelling approach and careful regularization.

2. Outlier Removal Results

Specifically, all observations outside the range of the lower cut-off threshold (Ist percentile) and upper cut-off threshold (99th percentiles) have been removed to create a data frame without outliers (df_no_outliers), but retaining the overall form of the central distribution.

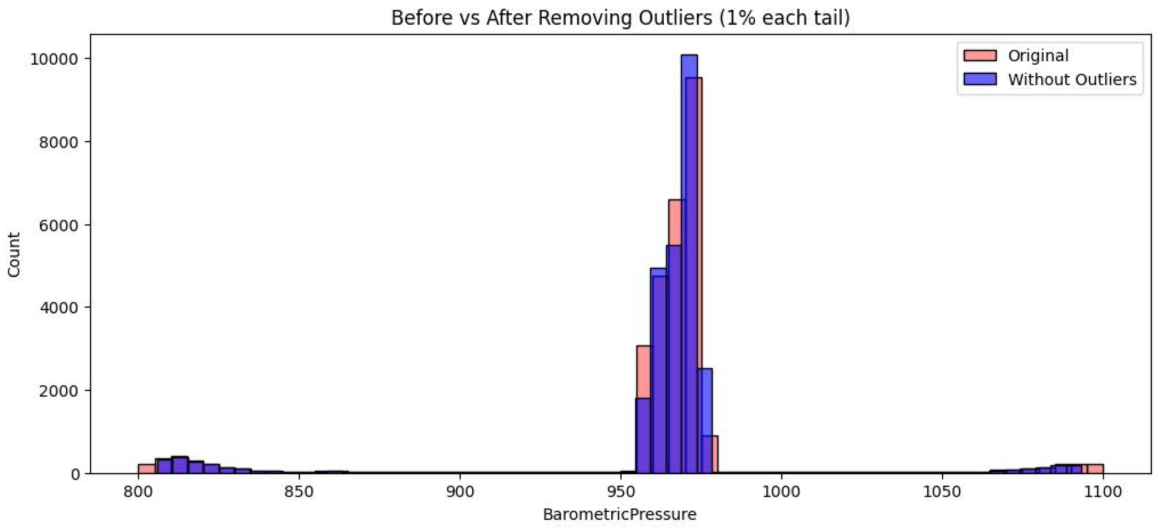


Figure 5.1: Outlier removal results.

Key Results:

- Dataset size reduced from 28446 to 27876 which means that 570 extreme readings were removed.
- The new summary statistics of the shape of BarometricPressure show that the standard deviation is lowered from values of 45.86 to 41.33.
- Notable difference in histogram plot of original BarometricPressure vs After outlier removal.

Most importantly, this step reduced actual variations while suppressing apparent spikes. Furthermore, it is able to yield a target distribution that is able to aid in regression learning with reduced noise and distortion.

3. Pearson Correlation Results

Here, BarometricPressure was tested for linear association with other sets of predictors by calculating Pearson's correlation coefficients (r) and their significance (p).

Feature	Pearson r	p-value
BarometricPressure_z	1	Not applicable (self-correlation)
Humidity	0.2418	$p < 1\times10^{-150}$
Humidity_z	0.2418	$p < 1\times10^{-150}$
AirTemperature_z	−0.1711	8.5×10^{-186}
AirTemperature	−0.1711	8.5×10^{-186}
WindDirection	0.1359	2.41×10^{-117}
HeatIndex_C	−0.1249	2.53×10^{-99}
Latitude	0.1209	4.02×10^{-93}
WindSpeed	0.1073	1.27×10^{-73}

Table 5.2: Pearson Correlation Results.

Interpretation of Results:

- Negative Coefficient Values: Air temperature and heat index exhibited inverse linear relationships.
- Positive Coefficient Values: Humidity showed the strongest positive linear relationship with BarometricPressure.

4. Spearman Correlation Results

Similarly to linear associations, monothonic associations of the target variable, BarometricPressure to other predictors sets of the dataset was calculated using spearman's coefficients and significance obtaining the results below:

Feature	**Spearman ρ**	**p-value**
BarometricPressure_z	1	Not applicable (self-correlation)
HeatIndex_C	−0.6017	$p < 1\times10^{-150}$
AirTemperature_z	−0.5981	$p < 1\times10^{-150}$
AirTemperature	−0.5981	$p < 1\times10^{-150}$
Humidity	0.1652	3.25×10^{-173}
Humidity_z	0.1652	3.25×10^{-173}
Latitude	0.0933	5.10×10^{-56}
Wind_v	0.0904	9.6×10^{-53}
WindDirection	0.0786	3.03×10^{-40}
Wind_u	−0.0657	1.43×10^{-28}

Table 5.3: Spearman Correlation Results.

Interpretation of Results:

- Heat Index and Air Temperature have a strong negative monothonic correlation relationship with BarometricPressure.
- Humidity showed the strongest positive monothonic relationship with BarometricPresure

The result shows that if humidity increases, Spearman correlation tells us that BarometricPressure tends to increase, but decreases when AirTemperature and HeatIndex increases regardless of the exact shape or linearity of the relationship. In other words, it measures whether the two variables move in the same direction (positive) or opposite directions (negative), even if the curve is non-linear.

5. Non-Linear Relationship Analysis (Mutual Information) Results

Here, By calculating Mutual Information (MI) scores, BarometricPressure was evaluated for non-linear or multi-dimensional associations between numerical predictors that might have been missed by linear association evaluation reproducing the results in table 5.4 below:

Feature	MI Score
BarometricPressure_z	8.774
HeatIndex_C	0.864
AirTemperature_z	0.826
AirTemperature	0.824
SRADCumulative	0.76
Humidity_z	0.607
Humidity	0.606
Latitude	0.44
SRAD_z	0.321
SRAD	0.321

Table 5.4: Mutual Information Results.

Result Interpretation:

- Temperature-driven variables dominate non-linear influence as seen that AirTemperature and HeatIndex show strong MI scores indicating a strong non-linear relationship with BarometricPressure.
- Radiation effects are non-linear contributors because the MI values of SRADCumulative and SRAD are significant.
- Humidity shows moderate non-linear influence.
- Latitude effects suggest spatial non-linearity. As the score seem ineffective when compared to temperature and radiation, latitude does indicate subtle geospatial gradients in the data.

6. Multicollinearity (Variance Inflation Factor) Results

Using Variance Inflation Factor (VIF) to measure how much the variance of each feature is inflated due to multicollinearity, provided a vital information for the selection of the final predictors form the cleaned dataset used to train the models.

Feature	VIF Score
HeatIndex_C	9.975474
SRADCumulative	2.545969
Hour	2.388847
Wind_v	1.915836
WindDirection	1.806153
Wind_u	1.583715
Latitude	1.435189
BarometricPressure_z	1.150583
Longitude	1.036161
Rain	1.015339
DayOfWeek	1.003877
GPSFix	1.001553

Table 5.5: Variance Inflation Factor Results.

Result Interpretation:

- All missing columns in the table above (WindSpeed, WindSpeed_z, Humidity, Humidity_z, AirTemperature, AirTemperature_z, SRAD, SRAD_z) have infinite VIFs indicating extremely high multicollinearity.
- Heat Index being a nonlinear combination of temperature and humidity exhibited high multicollinearity.
- Latitude, Longitude, Rain, DayOfWeek and GPSFix have VIF values close to 1, indicating that they do not exhibit multicollinearity with other variables.

In general, the final predictor sets excluded HeatIndex being that VIF score equals 9.98, higher that the maximum acceptable multicollinearity value of 9.

5.3 Notable Improvement Outcomes

The Improvement outcomes section evaluates the effectiveness of the implemented completeness framework by comparing the state of the raw dataset before and after the completion of the data completeness workflow. This evaluation extends to the performance of the three models: Random Forest Regressor, Decision Tree Regressor, and XGBoost Algorithm, to compare each of their performances on both raw and improved dataset.

1. Data Completeness Checks

Metrics	Column	Raw Dataset	Improved Dataset
Missing Values	All	0	0
Out-of-Range Values	BarometricPressure SRADCumulative	8,890 10,200	0 0
Data Inconsistency	CollectedDateAt	8,446	0
Data Inaccuracy	WindDirection	20	0
Data Duplication	CollectedDateAt	8	0
Outlier Detection	BarometricPressure	570	0
Data Indiversity	CollectedDateAt	28,446	0
Robustness to Noise	Wind_u Wind_v HeatIndex_C	0% 0% 0%	100% 100% 100%
Initial number of columns	All	12	24

Table 5.6: Data Completeness Checks Overview.

The comparison between the raw and improved datasets as depicted in Table 5.6 above shows a substantial enhancement in overall data quality. All major issues identified in the raw dataset—such as out-of-range values (e.g., 8,890 for BarometricPressure and 10,200 for SRADCumulative), 8,446 inconsistent timestamps, 20 inaccurate WindDirection values, and eight duplicated entries—were fully resolved in the improved dataset. Additionally, 570 detected outliers were handled using appropriate cleaning and correction techniques.

The improved dataset also addressed a major issue of data indiversity, correcting 28,446 irregular timestamp entries. Beyond error correction, the improved dataset introduced three new engineered features (Wind_u, Wind_v, HeatIndex_C), resulting in 100% robustness to noise for these components and expanding the dataset from 12 to 24 columns.

Overall, the improved dataset is more complete, consistent, and feature-rich, providing a significantly stronger foundation for machine learning analysis compared to the raw dataset.

2. Model Evaluation Comparison

Model	Dataset	Metric	Mean ± Std	Confidence Intervals
Random Forest	Raw	RMSE	15.6900 ± 0.7010	[14.316, 17.064]
	Raw	R^2	0.9272 ± 0.0068	[0.9139, 0.9405]
	Improved	RMSE	9.0904 ± 0.2960	[8.5102, 9.6706]
	Improved	R^2	0.9514 ± 0.0032	[0.9451, 0.9577]
Decision Tree	Raw	RMSE	18.3409 ± 0.6109	[17.1439, 19.5379]
	Raw	R^2	0.9006 ± 0.0067	[0.8875, 0.9137]
	Improved	RMSE	11.3313 ± 0.3337	[10.6773, 11.9853]
	Improved	R^2	0.9241 ± 0.0089	[0.9067, 0.9415]
XGBoost	Raw	RMSE	16.4225 ± 0.3769	[15.6845, 17.1605]
	Raw	R^2	0.9204 ± 0.0037	[0.9132, 0.9276]
	Improved	RMSE	9.1840 ± 0.5479	[8.111, 10.257]
	Improved	R^2	0.9504 ± 0.0042	[0.9422, 0.9586]

Table 5.7: Baseline Model Evaluation between Raw and Improved Dataset.

The comparison of machine learning models trained on the raw and improved datasets as seen in Table 5.7 above shows clear performance gains after applying the data completeness framework.
Across all three models—Random Forest, Decision Tree, and XGBoost—the RMSE decreased substantially, indicating more accurate predictions, while the R^2 scores increased, showing stronger explanatory power.

- Random Forest Regressor showed the largest improvement, with RMSE dropping from 15.690 to 9.090 and R^2 increasing from 0.927 to 0.951, confirming its robustness with cleaner data.

- Decision Tree Regressor also improved notably, with RMSE decreasing from 18.341 to 11.331 and R^2 increasing from 0.901 to 0.924.

- XGBoost demonstrated strong gains as well, reducing RMSE from 16.423 to 9.184 and raising R^2 from 0.920 to 0.950.

- The confidence interval (CI) provides a range of values within which the true population parameter is likely to lie with a certain level of confidence. The 95% confidence interval for each metric was calculated using CI = Mean ± (Z x Std), where Z equals 1.96.

In conclusion, the improved dataset consistently enabled higher accuracy and better generalization across all models, validating the effectiveness of the data cleaning and completeness enhancement process.

6 Discussion

This study aimed to design and implement a structured framework to enhance data completeness and evaluate its impact on machine learning model predictive performance. The findings clearly demonstrate that addressing data incompleteness such as missing values, placeholder entries, outliers and noise removal, out-of-range measurements, and duplicate records, substantially improves the overall dataset's quality.

The empirical results revealed that the models (Random Forest Regressor, Decision Tree Regressor and XGBoost) trained on the improved dataset consistently outperformed the models trained on the raw dataset. Reductions in their RMSE and R^2 values provide quantitative evidence of this improvement, highlighting how completeness-oriented data preprocessing reduces noise and strengthens the underlying relationships that machine learning models learn.

A notable contribution of this research is the integration of Cross-Industry Standard Process for Data Mining (CRISP-DM) into a systematic completeness framework. While existing studies frequently focused on isolated preprocessing steps, such as imputation, or feature scaling, this work combines multiple dimensions of completeness checks into a cohesive pipeline. The application of CRISP-DM multifaceted this approach, offering a more comprehensive solution compared to traditional preprocessing methods, which often overlook structural incompleteness within the data.

The implications of these findings extend to real-world applications, particularly in data-intensive sectors where decisions rely heavily on accurate and reliable data. Smart city infrastructures, such as the TUC SmartCityCloud platform, can benefit significantly from adopting such a completeness-driven data cleaning approach.

Despite these contributions, the study has certain limitations. Percentile-based winsorization was chosen over reconstruction-based imputation methods to prioritize data reliability, as extreme sensor values are more likely to indicate faults rather than valid measurements. Although approximately 2% of records were removed, this loss was considered acceptable given the substantial improvement in predictive accuracy. Future work may investigate advanced imputation techniques to assess whether greater data retention can be achieved without degrading model performance.

Additionally, Random K-fold cross-validation was applied under the assumption of weak temporal dependency; however, BarometricPressure is inherently time-dependent. Random splitting may therefore introduce temporal leakage risks and optimistic performance estimates. Future studies may consider chronological train–

test splits and time-series cross-validation strategies such as TimeSeriesSplit to ensure more realistic evaluation in forecasting scenarios.

The evaluation was conducted on a single dataset and focused primarily on tree-based machine learning models. Future research could extend this framework with different datasets or unsupervised learning algorithms capable of identifying hidden patterns of incompleteness.

In summary, this study underscores the fundamental importance of data completeness in machine learning workflows. The proposed framework enhances dataset reliability, strengthens feature–target relationships, and improves model performance.

7 Conclusion and Future Work

This research set out to investigate the role of data completeness in enhancing the predictive performance of machine learning models. The study introduced a structured and multi-dimensional data completeness workflow that integrates Initial data preprocessing, Exploratory Data Analysis (EDA) and model training. The implementation structurally applied rigorous data completeness checks and improvement measures to address missing values, unrealistic sensor readings, duplicated records, and non-compliant attribute relationships resulting to an improved Dataset. This improvements, emphasizing the effectiveness of data completeness in model performance, was reflected in significantly lower RMSE and R^2 values, for the Random Forest Regressor, Decision Tree Regressor and XGBoost compared to the results of the models on the raw dataset. However, Random Forest Regressor consistently delivered the strongest performance across both datasets.

While the research yielded promising results, several opportunities exist for further exploration and enhancement of the framework. First, the framework could adopt other error measuring metrics like Mean Squared Error (MSE) and Mean Absolute Error for better evaluation and comparisons of the different models performances. Additionally, the current evaluation is limited to a single dataset focused on atmospheric measurements. Applying the framework to additional larger datasets such as traffic, energy consumption, or health monitoring, would help validate its scalability, generalizability and robustness across domains.

The study employed univariate analysis and winsorization for outlier handling. Future efforts could incorporate more advanced anomaly detection methods such as isolation forests and Local Outlier Factor (LOF). These techniques may improve the identification of subtle or non-linear anomalies that traditional methods fail to capture.

While tree-based models were appropriate for the structured nature of the data, further research could include deep learning models for feature-rich or temporal datasets. Such an expanded evaluation would provide deeper insights into how different model families respond to improved data completeness. Additionally, synthetic benchmarking can be introduced to stress-test the proposed framework by generating artificial datasets with controlled missingness mechanisms (MCAR, MAR, and MNAR), enabling a more rigorous and systematic evaluation of its robustness in handling diverse and challenging data completeness scenarios.

Lastly, future research can extend the proposed framework to a streaming, online setting, enabling real-time detection of data completeness drift, such as sudden spikes in sensor dropouts or missing values, and automatically triggering alerts or model retraining when significant degradation is observed. In addition, federated completeness checks can be investigated at the network edge, allowing sensor nodes to locally analyze and validate data completeness prior to transmission, thereby reducing communication overhead, bandwidth consumption, and operational costs while improving overall system reliability.

In conclusion, this research highlights the substantial benefits of prioritizing data completeness within machine learning workflows. By establishing a structured and replicable approach to data quality improvement, the study contributes both methodological insights and practical tools for enhancing the reliability of AI systems. The future work directions outlined above provide promising opportunities to extend this framework and further advance the field of data-driven modeling and intelligent systems.

Bibliography

[1] O. H. Hamid, "From Model-Centric to Data-Centric AI: A Paradigm Shift or Rather a Complementary Approach?," 2022 8th International Conference on Information Technology Trends (ITT), Dubai, United Arab Emirates, 2022, pp. 196-199, doi: 10.1109/ITT56123.2022.9863935.

[2] Kreuzberger, Dominik, Niklas Kühl, and Sebastian Hirschl. "Machine learning operations (mlops): Overview, definition, and architecture." IEEE access 11 (2023): 31866-31879.

[3] Whang, S. E., Roh, Y., Song, H., & Lee, J. G. (2023). Data collection and quality challenges in deep learning: A data-centric ai perspective. The VLDB Journal, 32(4), 791-813. https://link.springer.com/article/10.1007/s00778-022-00775-9

[4] Hamid, Oussama. (2023). Data-Centric and Model-Centric AI: Twin Drivers of Compact and Robust Industry 4.0 Solutions. Applied Sciences. 13. 2753. 10.3390/app13052753.

[5] Van Dyk, David A., and Xiao-Li Meng. "The art of data augmentation." Journal of Computational and Graphical Statistics 10.1 (2001): 1-50.

[6] Zheng, H., Tian, B., Liu, X., Zhang, W., Liu, S., & Wang, C. (2022, August). Data Quality Identification Model for Power Big Data. In International Conference of Pioneering Computer Scientists, Engineers and Educators (pp. 20-29). Singapore: Springer Nature Singapore. https://link.springer.com/chapter/10.1007/978-981-19-5209-8_2

[7] Porjazovski, Dejan, Anssi Moisio, and Mikko Kurimo. "Out-of-distribution generalisation in spoken language understanding." arXiv preprint arXiv:2407.07425 (2024).

[8] Jia, Ruoxi, et al. "Towards efficient data valuation based on the shapley value." The 22nd International Conference on Artificial Intelligence and Statistics. PMLR, 2019.

[9] Pulicharla, Mohan Raja. "Data Versioning and Its Impact on Machine Learning Models." Journal of Science & Technology 5.1 (2024): 22-37

[10] Lopes, Lucas A., et al. "Automatic labelling of clusters of discrete and continuous data with supervised machine learning." Knowledge-Based Systems 106 (2016): 231-241.

[11] ClickData. Data Completeness. https://www.clicdata.com/blog/data-completeness/. Accessed: 2025-10-30.

[12] Metaplane. Data Completeness: Definition and Examples. https://www.metaplane.dev/blog/data-completeness-definition-examples. Accessed: 2025-10-30.

[13] A. Sidik, B. Jalius, S. N. A. Jawaddi, S. Zambri and A. Ismail, "Enhancing Data Quality Assessment Dashboard for Accuracy and Completeness Dimensions," 2024 5th International Conference on Computational Science & Information Management (ICoCSIM), Melbourne, Australia, 2024, pp. 109-115, doi: 10.1109/ICoCSIM65098.2024.00027.

[14] I. Taleb, M. A. Serhani and R. Dssouli, "Big Data Quality Assessment Model for Unstructured Data," 2018 International Conference on Innovations in Information Technology (IIT), Al Ain, United Arab Emirates, 2018, pp. 69-74, doi: 10.1109/INNOVATIONS.2018.8605945.

[15] R. Liu, G. Wang, W. H. Wang and F. Korn, "iCoDA: Interactive and exploratory data completeness analysis," 2014 IEEE 30th International Conference on Data Engineering, Chicago, IL, USA, 2014, pp. 1226-1229, doi: 10.1109/ICDE.2014.6816747.

[16] M. Huang, L. Li and P. Xuan, "Evaluating Data Consistency with Matching Dependencies from Multiple Sources," 2019 IEEE International Conference on Power Data Science (ICPDS), Taizhou, China, 2019, pp. 6-10, doi: 10.1109/ICPDS47662.2019.9017191.

[17] S. Mohammed, L. Budach, and M. Feuerpfeil, “The Effects of Data Quality on Machine Learning Performance,” *arXiv preprint*, 2022.

[18] S. Gurjar, “AI-Powered Predictive Analytics in Cloud-Based Insurance Systems,” *J. Inf. Syst. Eng. Manage.*, vol. 10, no. 1, 2025.

[19] Z. Yoon, C. Jordon, and T. Schroeder, “GAIN: Generative Adversarial Imputation Networks,” *arXiv preprint*, 2019.

[20] iceDQ. Data Reliability Engineered. https://icedq.com/6-data- quality-dimensions Accessed : 2025-11-04.

[21] D. Rao, V. N. Gudivada and V. V. Raghavan, "Data quality issues in big data," 2015 IEEE International Conference on Big Data (Big Data), Santa Clara, CA, USA, 2015, pp. 2654-2660, doi: 10.1109/BigData.2015.7364065.

[22] B. Ghojogh, A. Ghodsi, “Recurrent Neural Networks and Long Short-Term Memory Networks: Tutorial and Survey“. arXiv:2304.11461v1.

[23] Yu, Yong, Si, Xiaosheng, Hu, Changhua, and Zhang, Jianxun. A review of recurrent neural networks: LSTM cells and network architectures. Neural computation, 31 (7):1235–1270, 2019.

[24] A. V. Pindiyan and P. R.M, "Machine Learning-Based Imputation Techniques Analysis and Study," 2024 International Conference on Electrical Electronics and Computing Technologies (ICEECT), Greater Noida, India, 2024, pp. 1-6, doi: 10.1109/ICEECT61758.2024.10739330.

[25] N. Legapriyadharshini, T. Thirumalaikumari, K. Senbagam, R. Sarasu, H. A. Basha and P. Malathi, "Integration of AI and Cloud-Native Technologies for Personalized Mobile Banking Experiences," 2024 International Conference on Recent Innovation in Smart and Sustainable Technology (ICRISST), Bengaluru, India, 2024, pp. 1-6, doi: 10.1109/ICRISST59181.2024.10921880.

[26] J. Gao, C. Xie and C. Tao, "Big Data Validation and Quality Assurance -- Issuses, Challenges, and Needs," 2016 IEEE Symposium on Service-Oriented System Engineering (SOSE), Oxford, UK, 2016, pp. 433-441, doi: 10.1109/SOSE.2016.63.

[27] Eshaghi, Amir & aghaie, Abdollah. (2022). Data fusion techniques for fault diagnosis of industrial machines: a survey. 10.48550/arXiv.2211.09551.

[28] M. Abdallah, "Big Data Quality Challenges," 2019 International Conference on Big Data and Computational Intelligence (ICBDCI), Pointe aux Piments, Mauritius, 2019, pp. 1-3, doi: 10.1109/ICBDCI.2019.8686099.

[29] S. Deng et al., "Cloud-Native Computing: A Survey From the Perspective of Services," in Proceedings of the IEEE, vol. 112, no. 1, pp. 12-46, Jan. 2024, doi: 10.1109/JPROC.2024.3353855.

[30] Nishant Garg. *Apache kafka*. Packt Publishing Birmingham, UK, 2013.

[31] Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, DB Tsai, Manish Amde, Sean Owen, et al. Mllib: Machine learning in apache spark. Journal of Machine Learning Research, 17(34):1–7, 2016.

[32] Kalyan Sudhakar. Amazon web services (aws) glue. *International Journal of Management, IT and Engineering*, 8(9):108–122, 2018.

[33] Carbone, Paris & Katsifodimos, Asterios & Kth, † & Sweden, Sics & Ewen, Stephan & Markl, Volker & Haridi, Seif & Tzoumas, Kostas. (2015). Apache Flink™: Stream and Batch Processing in a Single Engine. IEEE Data Engineering Bulletin. 38.

[34] Apache Software Foundation. (2023). *Apache NiFi User Guide*. Retrieved from https://nifi.apache.org. Accessed: 2025-11-09.

[35] Spithourakis, G., & Roussos, G. (2020). Stream Processing Frameworks for IoT Data Quality. IEEE Internet of Things Journal, 7(5), 4293–4305.

[36] W. Y. Leong, Y. Z. Leong and K. R, "Generative AI Applications for Robust Shop Floor," 2025 International Conference on Cognitive Computing in Engineering, Communications, Sciences and Biomedical Health Informatics (IC3ECSBHI), Greater Noida, India, 2025, pp. 474-479, doi: 10.1109/IC3ECSBHI63591.2025.10991135.

[37] "IEEE Guide for an Architectural Framework for Explainable Artificial Intelligence," in IEEE Std 2894-2024 , vol., no., pp.1-55, 30 Aug. 2024, doi: 10.1109/IEEESTD.2024.10659410.

[38] H. Hong et al., "Visual Analytics System of Comprehensive Data Quality Improvement for Machine Learning using Data- and Process-driven Strategies," 2022 IEEE International Conference on Big Data (Big Data), Osaka, Japan, 2022, pp. 396-401, doi: 10.1109/BigData55660.2022.10020585.

[39] Ş. Usta, E. Güneş, Ü. İ. Deniz, U. Yıldız, K. Küçük and O. Akbulut, "Smart Data Annotation: Accelerating Data Labeling Processes through Deep Learning," 2025 10th International Conference on Computer Science and Engineering (UBMK), Istanbul, Turkiye, 2025, pp. 283-287, doi: 10.1109/UBMK67458.2025.11206861.

[40] V. S, G. Saravanan, K. S and M. K. A, "Data Governance in the Big Data Era: A Scalable Framework for Effective Management and Regulatory Compliance," 2024 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES), Chennai, India, 2024, pp. 1-5, doi: 10.1109/ICSES63760.2024.10910410.

[41] W. Chen, "Secure and Policy-Aware Serverless Pipelines for Automated Data Governance on Google Cloud," 2025 IEEE Secure Development Conference (SecDev), Indianapolis, IN, USA, 2025, pp. 92-93, doi: 10.1109/SecDev66745.2025.00020.

[42] K. Priyanka, S. Priyadharshini, N. A. Vignesh, A. Hameed, R. R. K and P. C, "Self-Healing Data Pipelines: Reinforcement Learning for Real-Time Fault Detection and Autonomous Recovery," 2025 International Conference on Metaverse and Current Trends in Computing (ICMCTC), Subang Jaya, Malaysia, 2025, pp. 1-4, doi: 10.1109/ICMCTC62214.2025.11196544.

[43] C. Bao, H. Zhao, R. Li and W. Xia, "Semi-Federated Learning Based on Autoencoder for Non-Intrusive Load Monitoring," 2023 IEEE 23rd International Conference on Communication Technology (ICCT), Wuxi, China, 2023, pp. 1069-1073, doi: 10.1109/ICCT59356.2023.10419689.

[44] S. Kumar, S. Datta, V. Singh, S. K. Singh and R. Sharma, "Opportunities and Challenges in Data-Centric AI," in IEEE Access, vol. 12, pp. 33173-33189, 2024, doi: 10.1109/ACCESS.2024.3369417.

[45] Y. Zhong, L. Wu, X. Liu, and J. Jiang, “Exploiting the potential of datasets: A data-centric approach for model robustness,” in Proc. AAAI Workshop Adversarial Mach. Learn., 2022.

[46] Kunal, M. Rana and J. Bansal, "The Future of OpenAI Tools: Opportunities and Challenges for Human-AI Collaboration," 2023 2nd International Conference on Futuristic Technologies (INCOFT), Belagavi, Karnataka, India, 2023, pp. 1-6, doi: 10.1109/INCOFT60753.2023.10424990.

[47] N. Seedat, F. Imrie and M. v. d. Schaar, "Navigating Data-Centric Artificial Intelligence With DC-Check: Advances, Challenges, and Opportunities," in IEEE Transactions on Artificial Intelligence, vol. 5, no. 6, pp. 2589-2603, June 2024, doi: 10.1109/TAI.2023.3345805.

[48] C. Krishnama, R. Puchhakayala, S. Kotha and F. Gouri, "Cost-Optimized Cloud Scheduling for ETL and Big Data Using AI," 2025 13th International Symposium on Digital Forensics and Security (ISDFS), Boston, MA, USA, 2025, pp. 1-6, doi: 10.1109/ISDFS65363.2025.11012004.

[49] M. T. Hayat Suhendar and Y. Widyani, "Machine Learning Application Development Guidelines Using CRISP-DM and Scrum Concept," 2023 IEEE International Conference on Data and Software Engineering (ICoDSE), Toba, Indonesia, 2023, pp. 168-173, doi: 10.1109/ICoDSE59534.2023.10291438.

[50] S. Maataoui, G. Bencheikh and G. Bencheikh, "Predictive Maintenance in the Industrial Sector: A CRISP-DM Approach for Developing Accurate Machine Failure Prediction Models," 2023 Fifth International Conference on Advances in Computational Tools for Engineering Applications (ACTEA), Zouk Mosbeh, Lebanon, 2023, pp. 223-227, doi: 10.1109/ACTEA58025.2023.10193983.

[51] Slater, L. J., Arnal, L., Boucher, M.-A., Chang, A. Y.-Y., Moulds, S., Murphy, C., Nearing, G., Shalev, G., Shen, C., Speight, L., Villarini, G., Wilby, R. L., Wood, A., & Zappa, M. (2023). *Hybrid forecasting: blending climate predictions with AI models.* Hydrology and Earth System Sciences, 27, 1865–1889. https://hess.copernicus.org/articles/27/1865/2023/

[52] H. C. Mandhare and S. R. Idate, "A comparative study of cluster based outlier detection, distance based outlier detection and density based outlier detection

techniques," 2017 International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2017, pp. 931-935, doi: 10.1109/ICCONS.2017.8250601.

[53] L. J. Slater et al., “Nonstationary weather and water extremes: a review of methods for their detection, attribution, and management,” Hydrol. Earth Syst. Sci., vol. 25, no. 7, pp. 3897–3935, Jul. 2021.

[54] Batini, C. and Scannapieco, M. (2016) Data Quality Dimensions. In: Data and Information Quality, Springer, Cham, 21-51. https://doi.org/10.1007/978-3-319-24106-7_2

[55] Sergiu Hart and Andreu Mas-Colell. Potential, value, and consistency. Econo metrica: Journal of the Econometric Society, pages 589–614, 1989

[56] BotPenguin. Exploratory data analysis - glossary, 2023. Accessed: 2025-11-23.

[57] Y. Shi et al., "Supporting Guided Exploratory Visual Analysis on Time Series Data with Reinforcement Learning," in IEEE Transactions on Visualization and Computer Graphics, vol. 30, no. 1, pp. 1172-1182, Jan. 2024, doi: 10.1109/TVCG.2023.3327200.

[58] C. Chatfield, The Analysis of Time Series: An Introduction, 6th ed. Boca Raton, FL, USA: Chapman and Hall/CRC, 1995.

[59] Kabita Sahoo, Abhaya Kumar Samal, Jitendra Pramanik, and Subhendu Kumar Pani. Exploratory data analysis using python. International Journal of Innovative Technology and Exploring Engineering, 8(12):4727–4735, 2019.

[60] V. Skliarov, O. Degtiarov, O. Zaporozhets, O. Letuchyi and V. Ievsieiev, "Utilizing of Univariate Analysis of Variance for Evaluation of Uncertainties Measurement Results of Properties of Reference Materials," 2022 XXXII International Scientific Symposium Metrology and Metrology Assurance (MMA), Sozopol, Bulgaria, 2022, pp. 1-4, doi: 10.1109/MMA55579.2022.9992863.

[61] G. B. Pinio Sinaga, E. Budhiarti Nababan and H. Mawengkang, "Combination of Local Outlier Factor and Winsorization for Clustering Outlier in Medical Records," 2023 11th International Conference on Information and Communication Technology (ICoICT), Melaka, Malaysia, 2023, pp. 110-114, doi: 10.1109/ICoICT58202.2023.10262730.

[62] DataScientest: Pearson and Spearman Correlations: A Guide to Understanding and Applying Correlation Methods. https://datascientest.com/en/pearson-and-spearman-correlations-a-guide-to-understanding-and-applying-correlation-methods. Accessed: 2025-11-24.

[63] T. Z. Htet and W. M. Oo, "Mutual Information Ratio-Based Approach for Rainfall Prediction with Multicollinearity," 2024 IEEE Conference on Computer Applications (ICCA), Yangon, Myanmar, 2024, pp. 1-6, doi: 10.1109/ICCA62361.2024.10532808.

[64] Bary, M.N.A., 2017. Robust regression diagnostic for detecting and solving multicollinearity and outlier problems: applied study by using financial data. Appl. Math. Sci. 11, 601–622.

[65] K. D. Pati, "Using Standard Error to Find the Best Robust Regression in Presence of Multicollinearity and Outliers," 2020 International Conference on Computer Science and Software Engineering (CSASE), Duhok, Iraq, 2020, pp. 266-271, doi: 10.1109/CSASE48920.2020.9142066. es (WRLTS)}.

[66] M. Abdar, M. Zomorodi-Moghadam and X. Zhou, "An Ensemble-Based Decision Tree Approach for Educational Data Mining," 2018 5th International Conference on Behavioral, Economic, and Socio-Cultural Computing (BESC), Kaohsiung, Taiwan, 2018, pp. 126-129, doi: 10.1109/BESC.2018.8697318.

[67] C. C. Kumar and V. Parthipan, "Performance Analysis of Predicting LIC Stock Price using Lasso Regression Compared with Random Forest Regression," 2024 Second International Conference Computational and Characterization Techniques in Engineering & Sciences (IC3TES), Lucknow, India, 2024, pp. 1-5, doi: 10.1109/IC3TES62412.2024.10877466.

[68] G. Li and H. Zhou, "Modeling and Estimation Methods for Student Achievement Recognition Based on XGBoost Algorithm," 2023 International

Conference on Evolutionary Algorithms and Soft Computing Techniques (EASCT), Bengaluru, India, 2023, pp. 1-6, doi: 10.1109/EASCT59475.2023.10392502.

[69] S. Yadav and S. Shukla, "Analysis of k-Fold Cross-Validation over Hold-Out Validation on Colossal Datasets for Quality Classification," 2016 IEEE 6th International Conference on Advanced Computing (IACC), Bhimavaram, India, 2016, pp. 78-83, doi: 10.1109/IACC.2016.25.

[70] K. Yang, H. Wang, G. Dai, S. Hu, Y. Zhang and J. Xu, "Determining the repeat number of cross-validation," 2011 4th International Conference on Biomedical Engineering and Informatics (BMEI), Shanghai, China, 2011, pp. 1706-1710, doi: 10.1109/BMEI.2011.6098566.

[71] Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2020). The M4 Competition: 100,000 forecasting methods and results. International Journal of Forecasting, 36(1), 54–74. https://doi.org/10.1016/j.ijforecast.2019.04.014.

[72] Zakaria El Mrabet, Niroop Sugunaraj, Prakash Ranganathan, and Shrirang Abhyankar. Random forest regressor-based approach for detecting fault location and duration in power systems. Sensors, 22(2):458, 2022.

[73] Leo Breiman. Random forests. Machine learning, 45:5–32, 2001.

[74] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in Proc. 22nd ACM SIGKDD Int.

[75] S. S. Mangun, Kusrini and M. Sulistiyono, "Predict Wildfires in Kalimantan using MODIS Products with Linear Regression, Gradient Boosting and Decision Tree algorithms," 2023 6th International Conference on Information and Communications Technology (ICOIACT), Yogyakarta, Indonesia, 2023, pp. 236-240, doi: 10.1109/ICOIACT59844.2023.10455944.

[76] Breiman, L., Friedman, J., Olshen, R.A., & Stone, C.J. (1984). Classification and Regression Trees (1st ed.). Chapman and Hall/CRC. https://doi.org/10.1201/9781315139470

[77] E. S. Saputra, A. G. Putrada and M. Abdurohman, "Selection of Vape Sensing Features in IoT-Based Gas Monitoring with Feature Importance Techniques,"

2019 Fourth International Conference on Informatics and Computing (ICIC), Semarang, Indonesia, 2019, pp. 1-5, doi: 10.1109/ICIC47613.2019.8985807.

[78] M. Mustaqeem, S. Mustajab and M. Alam, "Enhancing Software Defect Prediction Through Root Cause Analysis: A Hybrid Approach Integrating Permutation Importance with XGBoost (PERMBoost)," 2024 International Conference on Communication, Computer Sciences and Engineering (IC3SE), Gautam Buddha Nagar, India, 2024, pp. 61-67, doi: 10.1109/IC3SE62002.2024.10593406.

[79] W. Wang, F. Jiang and X. Wang, "MPCGNet: A Multiscale Feature Extraction and Progressive Feature Aggregation Network Using Coupling Gates for Polyp Segmentation," 2025 International Joint Conference on Neural Networks (IJCNN), Rome, Italy, 2025, pp. 1-8, doi: 10.1109/IJCNN64981.2025.11228888.

[80] D. Abadi et al. Scaling data validation in cloud environments. ACM Transac tions on Database Systems, 2020.

[81] X. Chu, I. F. Ilyas, S. Krishnan, and J. Wang. Data cleaning: Overview and emerging challenges. Proceedings of the VLDB Endowment, 9(12):1603–1614, 2016.

[82] W. Fan and F. Geerts. Foundations of Data Quality Management. Morgan & Claypool Publishers, 2012.

[83] H. M¨uller and B. Habegger. Data Consistency Challenges in Smart City Ap plications. Springer AI in Urban Planning, 2021.

[84] A. Chatfield, The Analysis of Time Series: An Introduction, 6th ed. Boca Raton, FL, USA: Chapman & Hall/CRC, 2003. (Widely cited for time-series completeness, missing time analysis, and temporal consistency in environmental data.)

[85] A. Yin, Y. Wang, J. Mao, H. Zhang and X. Chen, "Category-Contextual Relation Encoding Network for Few-Shot Object Detection," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 34, no. 9, pp. 8355-8367, Sept. 2024, doi: 10.1109/TCSVT.2024.3378978.

[86] Van Rossum, G. (1991). Python Tutorial. Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdam.

[87] Van Rossum, G., & Drake Jr., F. L. (2009). The Python Language Reference Manual. Network Theory Ltd.

[88] Millman, K. J., & Aivazis, M. (2011). “Python for Scientists and Engineers.” Computing in Science & Engineering, 13(2), 9–12.

[89] AI integration in simulation processes. https://nbn-resolving.org/urn:nbn:de:bsz:ch1-qucosa2-353861

[90] Cloud-based Online Solution for Automated Definition of Driving Routes for Verification of Automotive Systems. https://nbn-resolving.org/urn:nbn:de:bsz:ch1-qucosa2-353861

Chemnitzer Informatik-Berichte

In der Reihe der Chemnitzer Informatik-Berichte sind folgende Berichte erschienen:

CSR-24-01	Seyhmus Akaslan, Ariane Heller, Wolfram Hardt, Hardware-Supported Test Environment Analysis for CAN Message Communication, Juni 2024, Chemnitz
CSR-24-02	S. M. Rizwanur Rahman, Wolfram Hardt, Image Classification for Drone Propeller Inspection using Deep Learning, August 2024, Chemnitz
CSR-24-03	Sebastian Pettke, Wolfram Hardt, Ariane Heller, Comparison of maximum weight clique algorithms, August 2024, Chemnitz
CSR-24-04	Md Shoriful Islam, Ummay Ubaida Shegupta, Wolfram Hardt, Design and Development of a Predictive Learning Analytics System, August 2024, Chemnitz
CSR-24-05	Sopuluchukwu Divine Obi, Ummay Ubaida Shegupta, Wolfram Hardt, Development of a Frontend for Agents in a Virtual Tutoring System, August 2024, Chemnitz
CSR-24-06	Saddaf Afrin Khan, Ummay Ubaida Shegupta, Wolfram Hardt, Design and Development of a Diagnostic Learning Analytics System, August 2024, Chemnitz
CSR-24-07	Túlio Gomes Pereira, Wolfram Hardt, Ariane Heller, Development of a Material Classification Model for Multispectral LiDAR Data, August 2024, Chemnitz
CSR-24-08	Sumanth Anugandula, Ummay Ubaida Shegupta, Wolfram Hardt, Design and Development of a Virtual Agent for Interactive Learning Scenarios, September 2024, Chemnitz
CSR-25-01	Md. Ali Awlad, Hasan Saadi Jaber Aljzaere, Wolfram Hardt, AUTOSAR Software Component for Atomic Straight Driving Patterns, März 2025, Chemnitz
CSR-25-02	Billava Vasantha Monisha, Hasan Saadi Jaber Aljzaere, Wolfram Hardt, Automotive Software Component for QT Based Car Status Visualization, März 2025, Chemnitz
CSR-25-03	Zahra Khadivi, Batbayar Battseren, Wolfram Hardt, Acoustic-Based MAV Propeller Inspection, Mai 2025, Chemnitz

Chemnitzer Informatik-Berichte

Chemnitzer Informatik-Berichte

ISSN 0947-5125

Herausgeber: Fakultät für Informatik, TU Chemnitz
Straße der Nationen 62, D-09111 Chemnitz